CURSO PRÁTICO 47 DE PROGRAMAÇÃO DE COMPUTADORES

INPUT

PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

NOVA CULTURAL

Cz$ 30,00

INPUT

Vol. 4 Nº 47

NESTE NÚMERO

PROGRAMAÇÃO BASIC

MANCHETES E MAIS MANCHETES

Desenhe letras linha por linha. Como adaptar a escala dos caracteres para qualquer largura e altura. Desenhe suas próprias letras 921

PERIFÉRICOS

CANETAS ÓPTICAS

Como funciona uma caneta óptica. Sensibilidade. Resolução alta e baixa. Os diversos tipos. Problemas de compatibilidade..... 926

PROGRAMAÇÃO BASIC

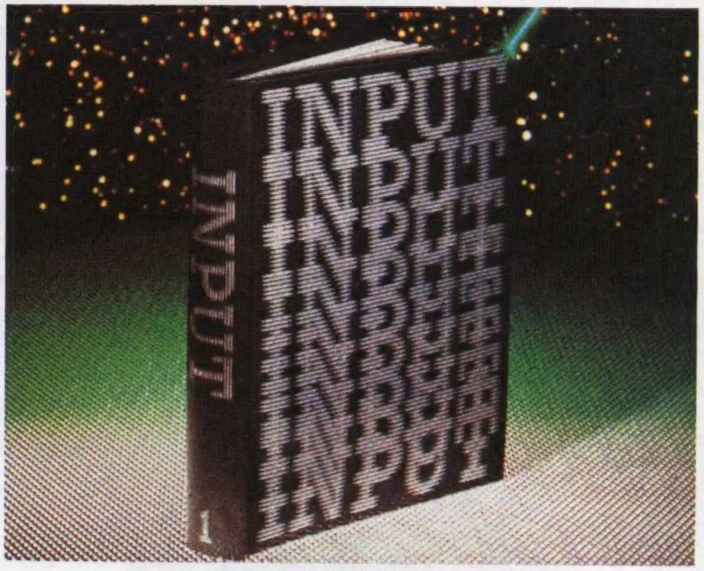
ACELERE SEUS PROGRAMAS

Velocidade de execução em programas BASIC. Cronometragem dos comandos. Como se relacionam estrutura, memória e velocidade..... 930

APLICAÇÕES

UM ASSISTENTE PARA O DOS

Funções do DOS. Um assistente inteligente. Como funciona o programa. Limitações 936



PLANO DA OBRA

"INPUT" é uma obra editada em fascículos semanais, e cada conjunto de 15 fascículos compõe um volume. A capa para encadernação de cada volume estará à venda oportunamente.

COMPLETE SUA COLEÇÃO

Exemplares atrasados, até seis meses após o encerramento da coleção, poderão ser comprados, a preços atualizados, da seguinte forma: 1. PESSOALMENTE — Por meio de seu jornaleiro ou dirigindo-se ao distribuidor local, cujo endereço poderá ser facilmente conseguido junto a qualquer jornaleiro de sua cidade. Em **São Paulo**, os endereços são: rua Brigadeiro Tobias, 773, Centro; av. Industrial, 117, Santo André; e no **Rio de Janeiro**: rua da Passagem, 93, Botafogo. 2. POR CARTA — Poderão ser solicitados exemplares atrasados também por carta, que deve ser enviada para DINAP — Distribuidora Nacional de Publicações — Números Atrasados — Estrada Velha de Osasco, 132, Jardim Teresa — CEP 06000 — Osasco — SP. Não envie pagamento antecipado. O atendimento será feito pelo reembolso postal e o pagamento, incluindo as despesas postais, deverá ser efetuado ao se retirar a encomenda na agência do Correio. 3. POR TELEX — Utilize o nº (011) 33 670 DNAP.
Em **Portugal**, os pedidos devem ser feitos à Distribuidora Jardim de Publicações, Lda. — Qta. Pau Varais, Azinhaga de Fetais — 2 685, Camarate — Lisboa; Apartado 57 — Telex 43 069 JARLIS P.
Atenção: Após seis meses do encerramento da coleção, os pedidos serão atendidos dependendo da disponibilidade do estoque.
Obs.: Quando pedir livros, mencione sempre título e/ou autor da obra, além do número da edição.

COLABORE CONOSCO

Encaminhe seus comentários, críticas, sugestões ou reclamações ao Serviço de Atendimento ao Leitor — Caixa Postal 9442, São Paulo — SP.

Editor
VICTOR CIVITA

REDAÇÃO
Diretor Editorial: Carmo Chagas

Editores Executivos: Antonio José Filho, Berta Sztark Amar

Editor Chefe: Paulo de Almeida
Editor de Texto: Cláudio A. V. Cavalcanti
Chefe de Arte: Carlos Luiz Batista
Assistentes de Arte: Dagmar Bastos Sampaio, Grace Alonso Arruda, Monica Lenardon Corradi
Secretária de Redação/Coordenadora: Stefania Crema
Secretários de Redação: Beatriz Hagström, José Benedito de Oliveira Damião, Maria de Lourdes Carvalho, Marisa Soares de Andrade, Mauro de Queiroz

COLABORADORES

Consultor Editorial Responsável: Dr. Renato M. E. Sabbatini (Diretor do Núcleo de Informática Biomédica da Universidade Estadual de Campinas)
Execução Editorial: DATAQUEST Assessoria em Informática Ltda., Campinas, SP
Tradução: Reinaldo Cúrcio
Tradução, adaptação, programação e redação: Abílio Pedro Neto, Aluísio J. Dornellas de Barros, Marcelo R. Pires Therezo, Raul Neder Porrelli
Coordenação Geral: Rejane Felizatti Sabbatini
Editora de Texto: Ana Lúcia B. de Lucena

COMERCIAL
Diretor Comercial: Roberto Martins Silveira
Gerente Comercial: Flávio Maculan
Gerente de Circulação: Denise Maria Mozol

PRODUÇÃO
Gerente de Produção: João Stungis
Coordenador de Impressão: Atílio Roberto Bonon

Preparador de Texto/Coordenador: Eliel Silveira Cunha
Preparadores de Texto: Alzira Moreira Braz, Ana Maria Dilguerian, Karina Ap. V. Grechi, Levon Yacubian, Luciano Tasca, Maria Teresa Galluzzi, Maria Teresa Martins Lopes, Paulo-Felipe Mendrone
Revisor/Coordenador: José Maria de Assis
Revisoras: Conceição Aparecida Gabriel, Isabel Leite de Camargo, Ligia Aparecida Ricetto, Maria de Fátima Cardoso, Nair Lúcia de Britto
Paste-up: Anastase Potaris, Balduino F. Leite, Edson Donato

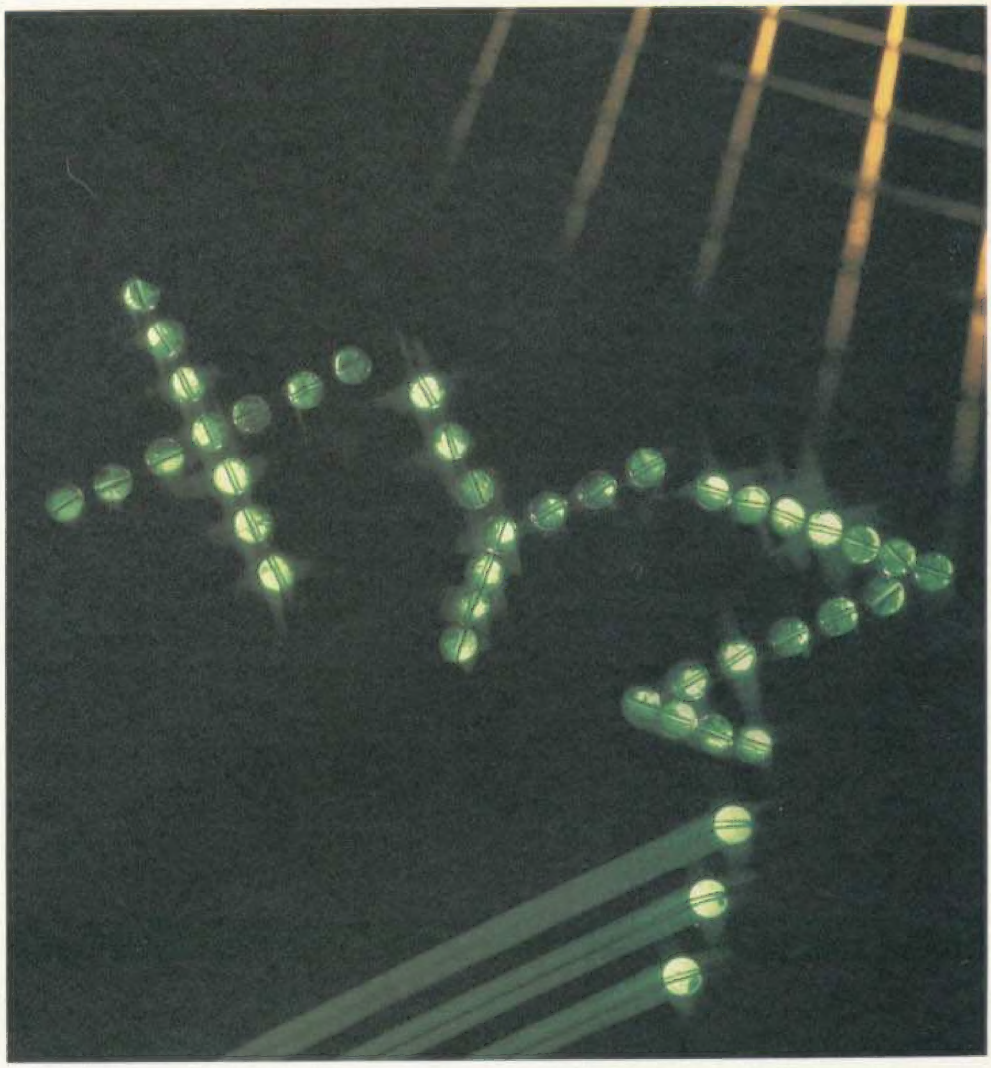
© Marshall Cavendish Limited, 1984/85.
© Editora Nova Cultural Ltda., São Paulo, Brasil, 1986.

Edição organizada pela Editora Nova Cultural Ltda.
Av. Brigadeiro Faria Lima, nº 2000 - 3º andar
CEP 01452 - São Paulo - SP - Brasil*
(Artigo 15 da Lei 5 988, de 14/12/1973).
Esta obra foi composta na AM Produções Gráficas Ltda. e impressa na Divisão Gráfica da Editora Abril S.A.

MANCHETES E MAIS MANCHETES

- DESENHE LETRAS LINHA POR LINHA
- ALTURA E LARGURA
- DESENHE SEU PRÓPRIO TIPO DE LETRA

Além de blocos gráficos e do conjunto de caracteres do computador, existem outros recursos que podem ser utilizados na criação de letras. Este artigo mostra um deles.



Os dois tipos de letra criados pelo programa do artigo *Manchetes e Letreiros* (página 912) são construídos a partir de blocos gráficos ou do conjunto de caracteres-padrão da máquina. Mas há um outro meio de criar letras: usando gráficos de alta resolução para desenhá-las, linha por linha.

O programa que se segue funciona de modo semelhante ao dos blocos gráficos. Cada letra é definida por uma série de dados alocados em uma instrução **DATA**, que informam ao computador como ela será. Essa informação é armazenada em uma matriz, e é possível dar entrada às palavras que se quer escrever na forma de um cordão alfanumérico. Como antes, as letras são montadas conforme as instruções encontradas e exibidas na tela. Mais tarde, você aprenderá a usar essas telas como parte de seus próprios programas.

UM TIPO A SEU GOSTO

Os gráficos de alta resolução facilitam enormemente a tarefa de criar o seu tipo de letra preferido. Isso acontece porque a expansão dos caracteres da ROM é limitada à duplicação da altura ou da largura dos caracteres. Ora, as letras constituídas por blocos gráficos estão completamente fixadas devido aos números das linhas **DATA**.

Por outro lado, os dados no nosso programa também fixam a maneira como a letra vai ser desenhada (o L, por exemplo, é formado de uma linha vertical e uma horizontal). Mas essas instruções são relativas e não determinam a aparência final das letras. Pense em cada letra como se ela estivesse encerrada dentro de uma caixa imaginária. Se a caixa é alta e estreita, temos uma letra alta e estreita; se ela for baixa e larga, assim será a letra. O programa é escrito de tal forma que é possível definir essas variáveis com fatores de escala.

Agora digite o programa...

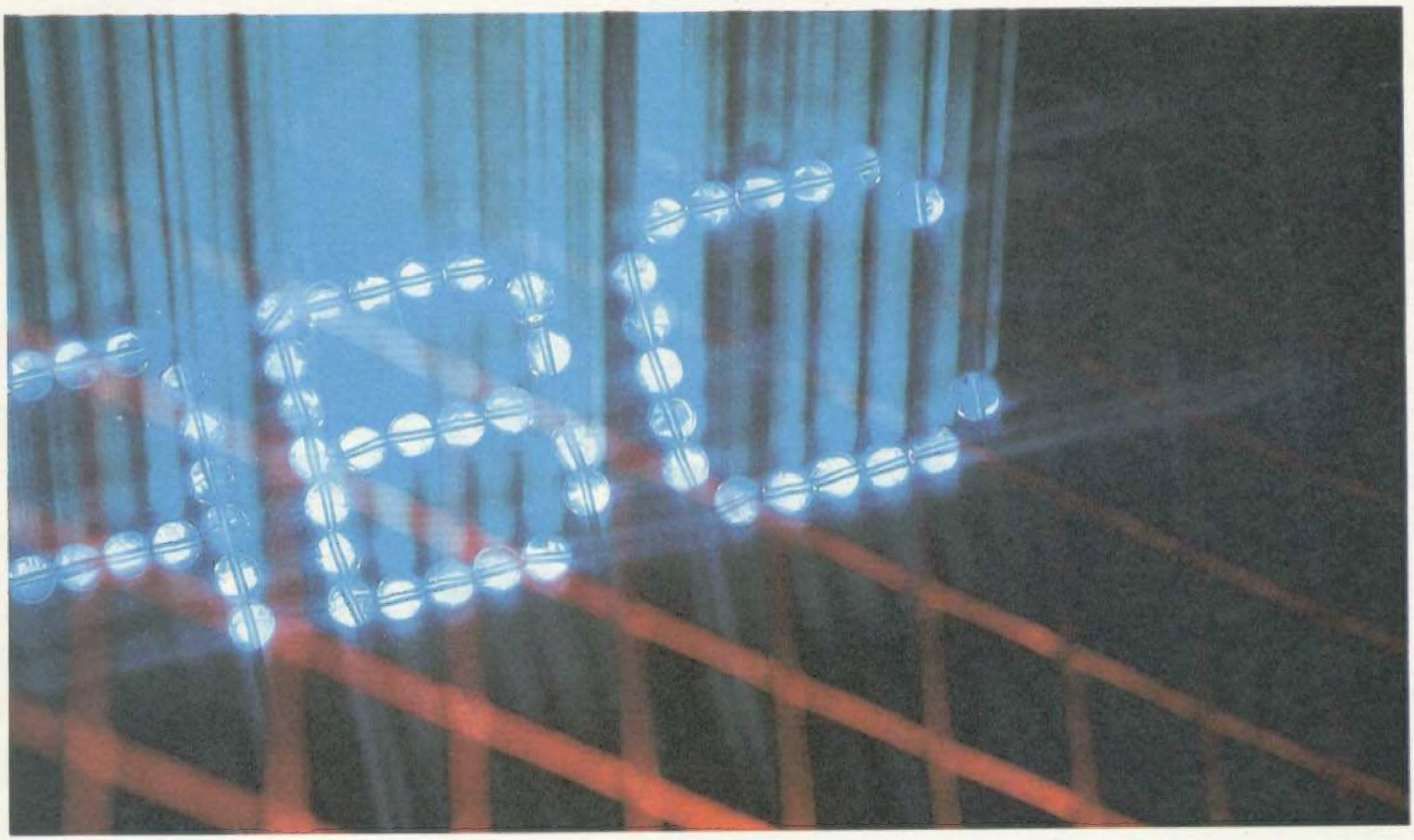
```
 10 POKE 23658,8
 20 DIM N(26): DIM A(26,12,2)
 30 FOR N=1 TO 26
 40 READ N(N)
 50 FOR M=1 TO N(N)
 60 READ A(N,M,1),A(N,M,2)
 70 NEXT M
 80 NEXT N
100 INPUT "INTRODUZA UMA PALAV
RA", LINE A$: IF A$="" THEN
GOTO 100
110 INPUT "DIGITE FATOR X",X
120 INPUT "DIGITE FATOR Y",Y
125 CLS
130 FOR N=1 TO LEN A$
140 LET T$=A$(N): IF T$<"A" OR
T$>"Z" THEN  NEXT N: GOTO 100
150 PLOT (10*(N-1)*X)+X*A(CODE
T$-55,1,1),20+Y*A(CODE T$-55,1
,2)
160 FOR M=2 TO N(CODE T$-55)
170 DRAW X*A(CODE T$-55,M,1),Y
*A(CODE T$-55,M,2)
180 NEXT M
190 NEXT N
200 GOTO 100
1000 DATA 8,0,0,5,1,1,4,0,1,-1,
0,-5,0,3,-6,0
1010 DATA 12,0,0,0,6,5,0,1,-1,0
,-1,-1,-1,-5,0,5,0,1,-1,0,-1,-1
```

```
,-1,-5,0
1020 DATA 8,6,1,-1,-1,-4,0,-1,1
,0,4,1,1,4,0,1,-1
1030 DATA 7,0,0,0,6,4,0,2,-2,0,
-2,-2,-2,-4,0
1040 DATA 7,6,0,-6,0,0,6,6,0,-6
,0,0,-3,5,0
1050 DATA 6,0,0,0,6,6,0,-6,0,0,
-3,5,0
1060 DATA 10,5,2,1,0,0,-1,-1,-1
,-4,0,-1,1,0,4,1,1,4,0,1,-1
1070 DATA 6,0,0,0,6,0,-3,6,0,0,
3,0,-6
1080 DATA 6,0,0,6,0,-3,0,0,6,-3
,0,6,0
1090 DATA 5,0,1,1,-1,4,0,1,1,0,
5
1100 DATA 6,0,0,0,6,0,-3,6,3,-6
,-3,6,-3
1110 DATA 3,6,0,-6,0,0,6
1120 DATA 5,0,0,0,6,3,-3,3,3,0,
-6
1130 DATA 4,0,0,0,6,6,-6,0,6
1140 DATA 9,1,0,-1,1,0,4,1,1,4,
0,1,-1,0,-4,-1,-1,-4,0
1150 DATA 7,0,0,0,6,5,0,1,-1,0,
-1,-1,-1,-5,0
1160 DATA 9,4,2,2,-2,-5,0,-1,1,
0,4,1,1,4,0,1,-1,0,-5
1170 DATA 9,0,0,0,6,5,0,2,-1,0,
-1,-1,-1,-5,0,4,0,2,-3
1180 DATA 12,0,1,1,-1,4,0,1,1,0
,1,-1,1,-4,0,-1,1,0,1,1,1,4,0,1
,-1
1190 DATA 4,3,0,0,6,-3,0,6,0
1200 DATA 6,0,6,0,-5,1,-1,4,0,1
,1,0,5
1210 DATA 3,0,6,3,-6,3,6
1220 DATA 5,0,6,1,-6,2,3,2,-3,1
,6
1230 DATA 5,0,0,6,6,-3,-3,-3,3,
6,-6
1240 DATA 5,3,0,0,3,-3,3,3,-3,3
,3
1250 DATA 4,6,0,-6,0,6,6,-6,0
```

```
10 PMODE 1,1:PCLS
20 DIM N(26),A(26,12,2)
30 FOR N=1 TO 26
40 READ N(N)
50 FOR M=1 TO N(N)
60 READ A(N,M,1),A(N,M,2)
70 NEXT M,N
80 CLS:INPUT"INTRODUZA A PALAVR
A ";A$:IF A$="" THEN GOTO 80
90 INPUT"QUAL O FATOR X ";X
100 INPUT"QUAL O FATOR Y ";Y
110 CLS:PCLS:SCREEN 1,0:FOR N=1
 TO LEN(A$)
120 T$=MID$(A$,N,1):IF T$<"A" O
R T$>"Z" THEN NEXT N:GOTO 80
130 J=(10*(N-1)*X)+X*A((ASC(T$)
-64),1,1):K=10+Y*A(ASC(T$)-64,1
,2)
140 FOR M=2 TO N(ASC(T$)-64)
150 F=X*A((ASC(T$)-64),M,1)
160 G=Y*A((ASC(T$)-64),M,2)
170 LINE(J,K)-(J+F,G+K),PSET
180 J=J+F:K=K+G
190 NEXT M,N
200 IF INKEY$="" THEN 200
210 GOTO 80
1000 DATA 9,0,6,0,-5,1,-1,4,0,1
,1,0,3,0,2,0,-3,-6,0
1010 DATA 12,0,0,0,6,5,0,1,-1,0
,-1,-1,-1,-5,0,5,0,1,-1,0,-1,-1
,-1,-5,0
1020 DATA 8,6,1,-1,-1,-4,0,-1,1
,0,4,1,1,4,0,1,-1
1030 DATA 7,0,0,0,6,4,0,2,-2,0,
-2,-2,-2,-4,0
1040 DATA 7,6,0,-6,0,0,6,6,0,-6
,0,0,-3,5,0
1050 DATA 7,0,0,6,0,-6,0,0,3,5,
0,-5,0,0,3
1060 DATA 10,7,1,-1,-1,-4,0,-1,
1,0,4,1,1,4,0,1,-1,0,-1,-1,0
1070 DATA 6,0,0,0,6,0,-3,6,0,0,
3,0,-6
1080 DATA 6,0,0,6,0,-3,0,0,6,-3
,0,6,0
1090 DATA 7,0,0,6,0,-3,0,0,5,-1
,1,-1,0,-1,-1
1100 DATA 6,0,0,0,6,0,-3,6,3,-6
,-3,6,-3
1110 DATA 3,0,0,0,6,6,0
1120 DATA 5,0,6,0,-6,3,3,3,-3,0
,6
1130 DATA 4,0,6,0,-6,6,6,0,-6
1140 DATA 9,1,0,-1,1,0,4,1,1,4,
0,1,-1,0,-4,-1,-1,-4,0
1150 DATA 7,0,6,0,-6,5,0,1,1,0,
1,-1,1,-5,0
1160 DATA 11,5,0,-4,0,-1,1,0,4,
1,1,4,0,1,-1,-2,-1,2,1,0,-4,-1,
-1
1170 DATA 9,0,6,0,-6,5,0,1,1,0,
1,-1,1,-5,0,4,0,2,3
1180 DATA 12,6,1,-1,-1,-4,0,-1,
1,0,1,1,1,4,0,1,1,0,1,-1,1,-4,0
,-1,-1
1190 DATA 4,3,6,0,-6,-3,0,6,0
1200 DATA 6,0,0,0,5,1,1,4,0,1,-
1,0,-5
1210 DATA 3,0,0,3,6,3,-6
1220 DATA 5,0,0,1,6,2,-3,2,3,1,
-6
1230 DATA 5,0,0,6,6,-3,-3,-3,3,
6,-6
1240 DATA 5,3,6,0,-3,-3,-3,3,3,
3,-3
1250 DATA 4,0,0,6,0,-6,6,6,0
```

```
10 SCREEN 0
20 DIM N(26),A(26,12,2)
30 FOR N=1 TO 26
40 READ N(N)
50 FOR M=1 TO N(N)
60 READ A(N,M,1),A(N,M,2)
70 NEXT M
75 NEXT N
80 INPUT"introduza a palavra";A
$
90 INPUT"introduza a escala do
x";X
100 INPUT"introduza a escala do
 y";Y
110 SCREEN 2
115 FOR N=1 TO LEN(A$)
120 T$=MID$(A$,N,1):IF T$<"A" O
R T$>"Z"THEN GOTO 80
130 J=(10*(N-1)*X)+X*A((ASC(T$)
-64),1,1):K=10+Y*A((ASC(T$)-64)
,1,2)
140 FOR M=2 TO N(ASC(T$)-64)
150 F=X*A((ASC(T$)-64),M,1)
160 G=Y*A((ASC(T$)-64),M,2)
170 LINE (J,K)-(J+F,K+G)
180 J=J+F:K=K+G
190 NEXT M
195 NEXT N
200 IF INKEY$="" THEN 200
210 GOTO 80
1000 DATA 9,0,6,0,-5,1,-1,4,0,1
,1,0,3,0,2,0,-3,-6,0
1010 DATA 12,0,0,0,6,5,0,1,-1,0
,-1,-1,-1,-5,0,5,0,1,-1,0,-1,-1
,-1,-5,0
1020 DATA 8,6,1,-1,-1,-4,0,-1,1
,0,4,1,1,4,0,1,-1
1030 DATA 7,0,0,0,6,4,0,2,-2,0,
-2,-2,-2,-4,0
1040 DATA 7,6,0,-6,0,0,6,6,0,-6
,0,0,-3,5,0
1050 DATA 7,0,0,6,0,-6,0,0,3,5,
0,-5,0,0,3
1060 DATA 10,7,1,-1,-1,-4,0,-1,
1,0,4,1,1,4,0,1,-1,0,-1,-1,0
1070 DATA 6,0,0,0,6,0,-3,6,0,0,
3,0,-6
1080 DATA 6,0,0,6,0,-3,0,0,6,-3
,0,6,0
1090 DATA 7,0,0,6,0,-3,0,0,5,-1
,1,-1,0,-1,-1
1100 DATA 6,0,0,0,6,0,-3,6,3,-6
,-3,6,-3
1110 DATA 3,0,0,0,6,6,0
1120 DATA 5,0,6,0,-6,3,3,3,-3,0
,6
1130 DATA 4,0,6,0,-6,6,6,0,-6
1140 DATA 9,1,0,-1,1,0,4,1,1,4,
0,1,-1,0,-4,-1,-1,-4,0
1150 DATA 7,0,6,0,-6,5,0,1,1,0,
1,-1,1,-5,0
1160 DATA 11,5,0,-4,0,-1,1,0,4,
1,1,4,0,1,-1,-2,-1,2,1,0,-4,-1,
-1
1170 DATA 9,0,6,0,-6,5,0,1,1,0,
1,-1,1,-5,0,4,0,2,3
1180 DATA 12,6,1,-1,-1,-4,0,-1,
1,0,1,1,1,4,0,1,1,0,1,-1,1,-4,0
,-1,-1
1190 DATA 4,3,6,0,-6,-3,0,6,0
1200 DATA 6,0,0,0,5,1,1,4,0,1,-
1,0,-5
1210 DATA 3,0,0,3,6,3,-6
1220 DATA 5,0,0,1,6,2,-3,2,3,1,
-6
1230 DATA 5,0,0,6,6,-3,-3,-3,3,
6,-6
1240 DATA 5,3,6,0,-3,-3,-3,3,3,
3,-3
1250 DATA 4,0,0,6,0,-6,6,6,0
```

```
5  LOMEM: 16384
10  HOME : HGR : HCOLOR= 3
20  DIM N(26),A(26,12,2)
30  FOR N = 1 TO 26
40  READ N(N)
50  FOR M = 1 TO N(N)
60  READ A(N,M,1),A(N,M,2)
```



```
70  NEXT M: NEXT N
75  HOME : HGR
80  VTAB 22: INPUT "INTRODUZA A
 PALAVRA ";A$: IF A$ = "" THEN
 GOTO 80
90  INPUT "QUAL O FATOR X? ";X
100  INPUT "QUAL O FATOR Y? ";Y

110  FOR N = 1 TO  LEN (A$)
120 T$ =  MID$ (A$,N,1): IF T$
< "A" OR T$ > "Z" THEN  NEXT N:
 GOTO 80
130 J = (10 * (N - 1) * X) + X
* A(( ASC (T$) - 64),1,1):K = 1
0 + Y * A( ASC (T$) - 64,1,2)
140  FOR M = 2 TO N( ASC (T$) -
 64)
150 F = X * A(( ASC (T$) - 64),
M,1)
160 G = Y * A(( ASC (T$) - 64),
M,2)
170  HPLOT J,K TO J + F,G + K
180 J = J + F:K = K + G
190  NEXT M: NEXT N
200  GET R$
210  GOTO 75
1000  DATA  9,0,6,0,-5,1,-1,4,0
,1,1,0,3,0,2,0,-3,-6,0
1010  DATA  12,0,0,0,6,5,0,1,-1
,0,-1,-1,-1,-5,0,5,0,1,-1,0,-1,
-1,-1,-5,0
1020  DATA  8,6,1,-1,-1,-4,0,-1
,1,0,4,1,1,4,0,1,-1
1030  DATA  7,0,0,0,6,4,0,2,-2,
0,-2,-2,-2,-4,0
1040  DATA  7,6,0,-6,0,0,6,6,0,
-6,0,0,-3,5,0
1050  DATA  7,0,0,6,0,-6,0,0,3,
5,0,-5,0,0,3
1060  DATA  10,7,1,-1,-1,-4,0,-
1,1,0,4,1,1,4,0,1,-1,0,-1,-1,0
1070  DATA  6,0,0,0,6,0,-3,6,0,
0,3,0,-6
1080  DATA  6,0,0,6,0,-3,0,0,6,
-3,0,6,0
1090  DATA  7,0,0,6,0,-3,0,0,5,
-1,1,-1,0,-1,-1
1100  DATA  6,0,0,0,6,0,-3,6,3,
-6,-3,6,-3
1110  DATA  3,0,0,0,6,6,0
1120  DATA  5,0,6,0,-6,3,3,3,-3
,0,6
1130  DATA  4,0,6,0,-6,6,6,0,-6

1140  DATA  9,1,0,-1,1,0,4,1,1,
4,0,1,-1,0,-4,-1,-1,-4,0
1150  DATA  7,0,6,0,-6,5,0,1,1,
0,1,-1,1,-5,0
1160  DATA  11,5,0,-4,0,-1,1,0,
4,1,1,4,0,1,-1,-2,-1,2,1,0,-4,-
1,-1
1170  DATA  9,0,6,0,-6,5,0,1,1,
0,1,-1,1,-5,0,4,0,2,3
1180  DATA  12,6,1,-1,-1,-4,0,-
1,1,0,1,1,1,4,0,1,1,0,1,-1,1,-4
,0,-1,-1
1190  DATA  4,3,6,0,-6,-3,0,6,0

1200  DATA  6,0,0,0,5,1,1,4,0,1
,-1,0,-5
1210  DATA  3,0,0,3,6,3,-6
1220  DATA  5,0,0,1,6,2,-3,2,3,
1,-6
1230  DATA  5,0,0,6,6,-3,-3,-3,
3,6,-6
1240  DATA  5,3,6,0,-3,-3,-3,3,
3,3,-3
1250  DATA  4,0,0,6,0,-6,6,6,0
```

Os programas aqui apresentados para as diferentes linhas de microcomputadores são muito semelhantes entre si. Em todos eles vamos encontrar 26 declarações **DATA**, uma para cada letra do alfabeto. Essas declarações contêm uma série de números que dizem ao computador como desenhar a letra, utilizando pequenas linhas.

O primeiro número da lista dá o total de linhas que entram na composição de cada letra — um L, por exemplo, requer menos linhas que um S. O número máximo usado é doze.

Os números seguintes são arranjados em pares, dando as coordenadas x e y de cada pequena seção da linha. Como já foi dito, tais números são relativos; assim, as linhas que serão efetivamente desenhadas dependem de fatores de escala. O primeiro par de números especifica o ponto de início da letra dentro de nossa caixa imaginária.

Os números contidos nas instruções **DATA** do programa são lidos para duas matrizes, **N** e **A**. **A** é uma matriz tridimensional; de 26 (número de letras) por 12 (número máximo de linhas) por 2 (vetores x e y).

COMO ESCOLHER UM CORDÃO

As linhas que permitem a entrada de palavras são semelhantes às do programa do artigo *Manchetes e Letreiros*. Elas verificam a entrada para evitar um cordão nulo; não existe, neste caso, nenhum limite para o número de caracteres utilizados.

Em seguida, o programa pede dois valores — um fator **X** e um fator **Y**. Esses valores determinam as reais dimensões de cada letra. Grosso modo, a escala 1 representa o tamanho padrão do caractere; 2 fornece altura ou largura dupla, enquanto 0.5, por sua vez, oferece metade do tamanho. Para valores menores que a unidade, podem-se obter efeitos estranhos. Uma vez que o computador não pode traçar uma fração de pixel, ele é obrigado a desenhar o pixel todo. Letras com um maior número de linhas apresentam mais facilidade para que isso ocorra. Desse modo, um S, por exemplo, pode terminar maior quando comparado com um T.

Se atribuirmos diferentes valores para cada fator, poderemos produzir interessantes variações nos caracteres, criando assim letras altas e magras, baixas e gordas etc.

Como não existe limite para o tamanho do cordão, é necessário tomar todo o cuidado no sentido de que o número de letras não ultrapasse as dimensões da tela. Quando isso ocorre, o computador emite uma mensagem de erro.

O cálculo de quantos caracteres cabem na tela para cada fator de escala é relativamente simples. Note que o valor importante é o fator X, que determina a largura da letra. Se você tiver um fator X igual a 2, só poderá colocar a metade do que seria possível com o fator 1. Com um fator 4, caberá apenas 1/4 dos caracteres.

ANALISE O CORDÃO

Assim como os outros programas que criavam letras, este usa um laço para verificar cada caractere do cordão. No Spectrum ele começa na linha 130; no TRS-Color, na linha 110.

O programa continua de maneira similar ao gerador de caracteres anterior, montando um cordão **T$** exatamente igual ao cordão original. Ao mesmo tempo, ele verifica se o caractere é uma letra de **A** a **Z**; qualquer outro caractere é tratado como se fosse um espaço. Se não for, o computador executará um **NEXT** para fazer avançar o laço **FOR...NEXT**. Se não houver mais letras, o programa voltará para que você possa dar entrada a mais palavras.

A ROTINA DE DESENHO

Quando o computador encontra uma letra no cordão, vai para a linha 150 (no Spectrum), que dá início à rotina de desenho. Nos outros computadores ela está na linha 170. O conteúdo dessa linha, porém, difere de computador para computador, pois cada um deles tem um tamanho diferente de tela e conta com comandos distintos para desenhar em alta resolução.

A base para a rotina, entretanto, é a mesma para todos os micros. A variável de controle para o laço **FOR...NEXT**, **N**, é usada como um guia para a coordenada x da posição inicial; a ela são adicionados dois valores pelo computador. O primeiro é um valor relativo — a coordenada para desenho da matriz **A** —, enquanto o segundo é um valor que leva em conta o tamanho da tela e os fatores de escala.

Você já viu que um dos três elementos dessa matriz é usado para separar os detalhes das coordenadas x e y. Os outros dois elementos formam cada uma das 26 letras do alfabeto com até 24 números (lembre-se de que doze é o número máximo de linhas usadas por letra). A matriz **N** diz ao computador quantas linhas devem ser empregadas na definição de cada letra. Assim, quando o computador vai desenhar o caractere A, por exemplo, ele verifica inicialmente quantas linhas devem ser traçadas na matriz **N**. Esse número controla o laço para desenhar a quantidade exata de linhas de cada letra.

Os valores dos números contidos em **A**, a matriz principal do programa (tridimensional), equivalem ao código de caractere de cada letra menos 64 — esta é a razão pela qual as linhas 130 a 180 contêm expressões como **ASC(T$)-64** ou **CODE T$-64** ou **A-64**. Isso significa que o computador pode tomar o código de uma letra e usá-lo para contar linhas de desenho. Os valores relativos para x e y guardados na matriz são então multiplicados pelo fator de escala que você escolheu e passados para os comandos de desenho.

Assim que acaba de desenhar uma letra, o computador passa para a seguinte (caso haja) no cordão e o processo continua. Se as letras acabarem, o computador volta para a linha 80 (no Spectrum) ou 100 (nos outros micros) para que se possa entrar um novo cordão. Os microcomputadores das linhas TR-Color e MSX esperam que você tecle alguma coisa antes, visto que a tela de alta resolução tem que ser apagada para dar lugar à tela de texto.

Eis aqui as três versões para novas letras no Spectrum.

DESENHE SUAS PRÓPRIAS LETRAS

Você já viu três exemplos de tipos de letra. Inspirando-se neles, crie à vontade seus próprios caracteres. Os caracteres da ROM só podem ser expandidos, mas nada impede que as rotinas expansoras sejam utilizadas para trabalhar com seus caracteres redefinidos — uma explicação de como isso pode ser feito foi dada anteriormente no artigo *Conjunto de Blocos Gráficos* (1), à página 526.

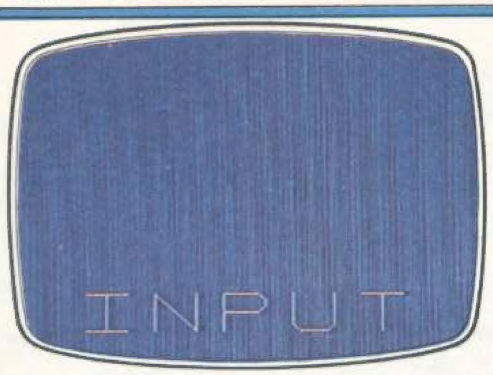
As variações com os outros dois métodos são praticamente infinitas: você pode desenhar seus próprios caracteres a partir de blocos gráficos e guardá-los em uma matriz, como no programa do artigo *Manchetes e Letreiros*. Se não estiver satisfeito com o resultado, tente redesenhar tudo. Outra opção consiste em alterar a aparência das letras deste artigo, modificando os valores das linhas **DATA**.

USE OS NOVOS CARACTERES

Não é difícil criar letras maiores para serem mostradas na tela; mas, a não ser que se possa usá-las em algum programa, não há grande vantagem em se fazer isso. Se você optar pelas letras criadas neste programa ou no anterior, procure empregá-las em seus próprios programas.

Isso pode ser feito de várias maneiras. A mais comum incorpora o programa gerador de letras como uma sub-rotina do seu programa e chama-o sempre que for necessário.

Se você quiser usar o expansor de caracteres, essa será provavelmente a melhor solução. Mas, para o programa



Alterando-se os fatores X e Y, as letras podem ser traçadas em qualquer tamanho.

As telas mostram os tamanhos normal, extra-largo e extra-alto.

atual, ela está longe de ser razoável, uma vez que o programa toma muito espaço da memória e pouco sobra para seu próprio programa.

Como você verá em seguida, esse problema pode ser resolvido de várias maneiras: todas elas utilizam o programa para desenhar as letras, que são armazenadas de modo a estar disponíveis para uso posterior.

ARMAZENAGEM DOS DESENHOS

Uma dessas maneiras consiste em armazenar seus desenhos em gráficos UDG. Isso é fácil de fazer no TRS-Color, já que este possui o comando **GET**, que guarda o que está na tela. Embora com maiores dificuldades, o mesmo efeito pode ser obtido, em outros computadores, por meio do comando **POKE**.

É possível ainda gravar a parte da memória que contém os gráficos, como um bloco de código de máquina. Mais tarde, pode-se carregar toda a tela de volta para a memória. Entretanto, enquanto ela estiver na memória do micro, é preciso protegê-la, colocando-a acima ou abaixo do BASIC (para maiores detalhes, veja o artigo da página 526). Uma vez na memória, ela está em condições de ser utilizada.

A melhor maneira de fazer isso é escrever uma rotina para ler o código da área protegida da RAM e passá-lo para a área de tela; ou, ainda, alterar o apontador do conjunto de caracteres, de modo que seu código se transforme em vários caracteres que poderão ser impressos. A rotina para mover o bloco de código da RAM para a área de tela seria algo como:

```
1000 FOR X=1 TO (COMPRIMENTO DO
 BLOCO DE CODIGO)
1010 POKE (ENDERECO DE INICIO D
A MEMORIA DE TELA)+X,PEEK (ENDE
RECO DE INICIO DO BLOCO DE CODI
GO)+X
1020 NEXT X
```

Esta, porém, não é a melhor solução para o problema, uma vez que levaria longo tempo para ser concluída. A dificuldade, felizmente, pode ser contornada por meio de uma rotina em linguagem de máquina que se encarregue de executar esse trabalho.

COMO GRAVAR A TELA

Uma solução alternativa consiste em gravar a tela e carregá-la juntamente com o programa. Nesse caso, o desenho permaneceria na tela até que você se decidisse a apagá-lo. Na maioria dos microcomputadores, esta constitui, sem dúvida, a melhor opção.

S

O Spectrum tem um comando especial que permite gravar em fita uma tela gráfica com grande facilidade. O desenho é gravado como um bloco de memória — processo descrito nas páginas 574 e 575; neste caso, porém, não é necessário especificar o endereço inicial e o tamanho do bloco. Basta usar **SCREEN$** logo após o nome de arquivo. Digitado no modo direto, esse comando grava uma figura com o nome "pic".

```
SAVE "pic" SCREEN$
```

Para carregar a figura:

```
LOAD "" SCREEN$
```

Pode-se, por exemplo, usar a tela como título para um jogo. Carrega-se primeiro a tela e a seguir o programa. Assim, ela ficará visível enquanto o programa estiver sendo carregado:

```
10 LOAD "" SCREEN$
20 LOAD "JOGO"
```

Grave agora com o comando:

```
SAVE "CARREG" LINE10
```

de modo que ele será executado automaticamente. O jogo será gravado com

O BASIC é muito lento e eu não gosto de programar em código de máquina. Tenho outras opções?

Existe uma família de linguagens que, digamos, estão "a meio caminho". Elas são compiladas, e não interpretadas. Como vimos anteriormente, o que toma tempo no BASIC é a interpretação de cada comando toda vez que o programa é rodado.

As linguagens compiladas usam um outro tipo de programa — parecido com os Assembler empregados na programação em código de máquina — que converte a linguagem de alto nível para código de máquina. Assim, depois da compilação, estaremos rodando um programa em código de máquina, e não em linguagem de alto nível.

```
SAVE "JOGO" LINE10
```

de modo que ele será executado automaticamente também. É importante lembrar que os programas têm que ser gravados na ordem correta na fita.

T

Para gravar uma tela em fita, utilize o comando:

```
CSAVEM"nomearquivo",1536,7679,
35725
```

Os primeiros dois números são os endereços inicial e final da área de gráficos (página 1 de qualquer **PMODE**, a não ser que se tenha um drive). **CLOADM** serve para carregar a figura de volta para a memória.

Essa tela gravada pode ser usada como uma página-título para um jogo; para isso, é preciso adicionar uma rotina de carregamento no início:

```
1 PMODE1,1:PCLS:SCREEN1,0
2 CLOADM
3 FOR D=1 TO 1000:NEXT
```

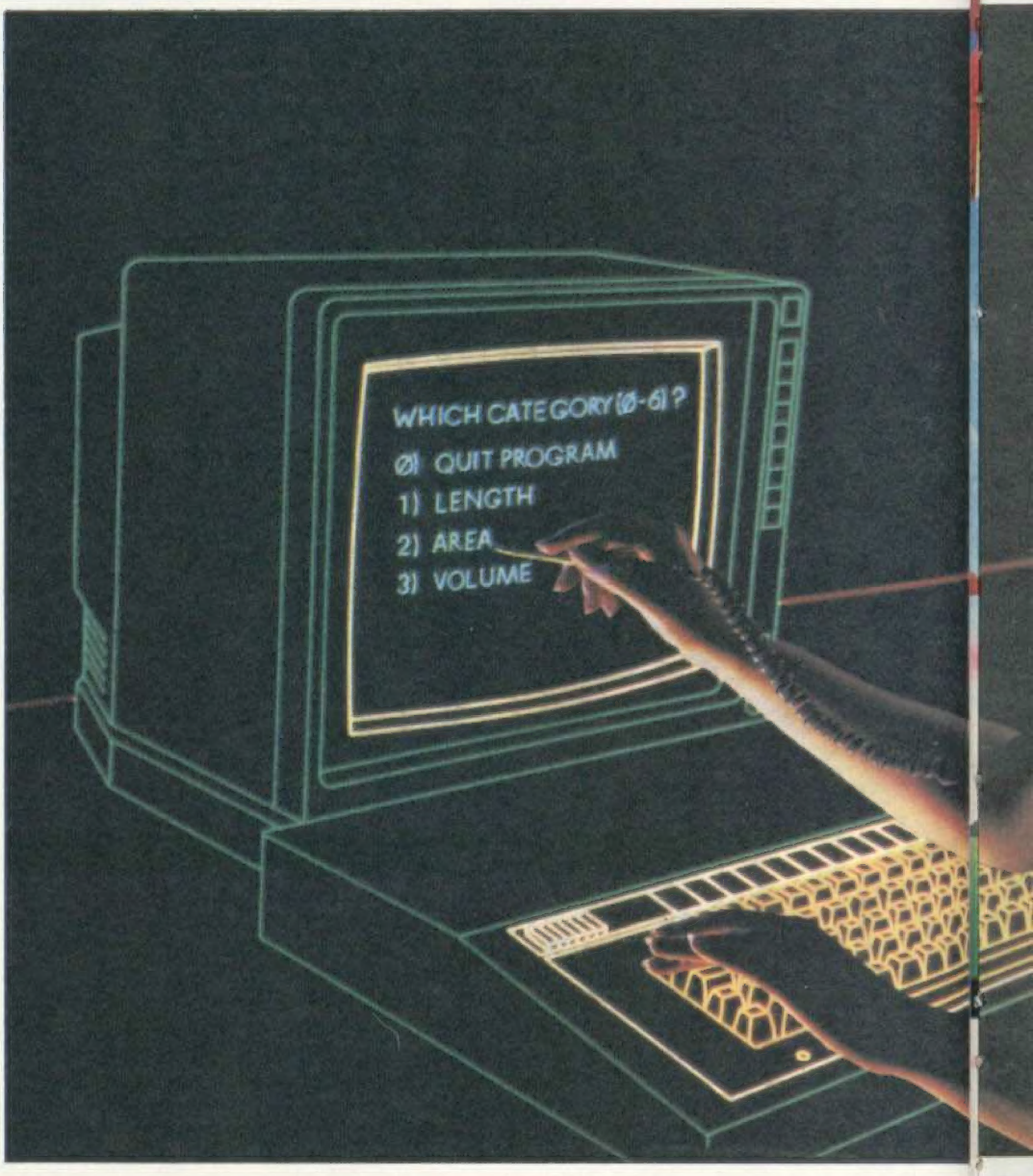
e certificar-se de que a figura será gravada logo após o jogo. Isso funcionará enquanto o programa não reduzir o número de páginas gráficas disponíveis inicialmente (neste caso, a figura poderia sobrepor-se ao programa).

CANETAS ÓPTICAS

Espécie muito particular de ponteiro eletrônico, as canetas ópticas servem para fazer gráficos na tela e para selecionar itens de um menu. Veja como trabalhar com elas.

Comunicar-se com um computador geralmente significa recorrer ao teclado para digitar informações. Esta, porém, não é a única forma de comunicação com a máquina. Além dela, existe uma série de métodos alternativos. Provavelmente, o primeiro lugar dessa série pertence ao joystick, que pode ser usado tanto em jogos como em aplicações mais "sérias". Outro método igualmente barato e que apresenta nítida vantagem é o da caneta óptica.

As canetas ópticas, como o próprio nome sugere, têm grande semelhança com uma caneta comum, pois funcionam com base em movimentos de "desenhar" ou de apontar para coisas sobre a tela. Como são capazes de detectar o ponto da tela para o qual estão sendo apontadas, elas têm uma enorme série de aplicações potenciais, como, por exemplo, selecionar itens de um menu mostrado na tela e acionar as funções de um programa. Caso fosse utilizar o teclado em situações como essas, o usuário seria obrigado a afastar os olhos da tela e pressionar uma ou mais teclas. Com uma caneta óptica, porém, tudo fica bem mais rápido e simples: basta apontar para a opção.



APLICAÇÕES

As canetas ópticas são particularmente apropriadas para aquelas aplicações em que o usuário precisa localizar rapidamente pontos espalhados pela tela, como quando deseja entrar as coordenadas de um ou mais pontos, colocar uma marca, desenhar uma reta, selecionar um menu etc.

Precisas, rápidas e de fácil manipulação, as canetas ópticas se prestam muito bem a aplicações gráficas. Nesse tipo de aplicação, o emprego das teclas para controle do cursor gráfico (ou, pior ainda, a utilização das teclas numéricas para entrar coordenadas) implicaria num lento e laborioso processo. Já com uma caneta óptica pode-se desenhar literalmente à mão livre: você traça apenas os contornos com a caneta, e o resultado aparece diretamente na tela. Outra técnica consiste em compor um desenho, usando a caneta para apontar para um menu com símbolos gráficos, cores, sombreados etc.

A caneta óptica mostra-se também de grande valia em programas de aplicações mais sérias, como processamento de textos, ou contabilidade (do tipo "você obtém o que vê"). Isso significa que, em vez de entrar instruções pelo teclado, elas são entradas por intermédio de uma caneta óptica que aponta para as opções disponíveis.

Até recentemente, programas aplicativos que empregassem canetas ópticas eram desenvolvidos pelos próprios usuários. Atualmente, com o crescente interesse por esses periféricos, algumas empresas produtoras de software já estão oferecendo programas que usam especificamente a caneta óptica.

Alguns programas "artísticos" ou de projeto por computador possibilitam a criação de desenhos à base de figuras

- UMA ALTERNATIVA PARA O TECLADO
- O QUE SE PODE FAZER COM UMA CANETA ÓPTICA?
- COMO FUNCIONA
- SENSIBILIDADE
- RESOLUÇÃO ALTA OU BAIXA?
- ESCOLHA UMA CANETA ÓPTICA
- COMPATIBILIDADE
- OUTROS PROBLEMAS

geométricas (retas, círculos, elipses etc.), que podem ser preenchidas com cores. Muitos desses programas permitem também o uso de canetas ópticas para desenhos à mão livre sobre a tela. Com eles é possível elaborar trabalhos complexos e criativos.

Esse processo funciona da seguinte forma: um dos programas utiliza a caneta óptica para definir símbolos ou caracteres gráficos. Um segundo programa aproveita esses símbolos em projetos mais complexos. Aqui, a caneta óptica oferece nítidas vantagens sobre outros métodos de entrada gráfica e de seleção de opções.

Já é possível encontrar no mercado jogos que recorrem a canetas ópticas para que o jogador transporte figuras de um ponto para outro da tela. O xadrez e outros jogos de tabuleiro são bons exemplos desse emprego.

Palavras cruzadas, jogos de cartas, labirintos e quebra-cabeças variados completam o quadro de atividades lúdicas em que as canetas ópticas têm funções importantes. Essas funções são extensivas a jogos de aventuras, em que o jogador pode apontar para os objetos que deseja apanhar ou levar.

As canetas ópticas podem ainda assumir formas especiais, como o rifle óptico que serve para centrar a mira em alvos na tela. Esse rifle já existe tanto em micros domésticos quanto em certos videogames. Está, portanto, especificamente voltado para jogos.

COMO FUNCIONA A CANETA ÓPTICA

Uma caneta óptica é usada basicamente para detectar pontos luminosos presentes na tela por meio de um fotodetector. Quando ela é apontada para alguma locação na tela, ocorre um trabalho conjunto entre os circuitos eletrônicos de detecção na caneta e o software carregado no computador; o objetivo desse trabalho é calcular as coordenadas do ponto para o qual a caneta está sendo apontada.

Uma das técnicas desse processo baseia-se no modo como a imagem é produzida em um cinescópio, tubo de TV ou monitor. O feixe de elétrons que bombardeia a tela é movimentado de forma a deslocar um pequeno ponto de luz sobre ela em um padrão que vai da esquerda para a direita e de cima para baixo. Esse padrão tem cerca de seiscentas linhas de varredura, repetidas a cada 1/60 de segundo. Como a varredura é muito rápida, o olho humano percebe apenas uma imagem estável.

A caneta óptica tem na ponta um minúsculo sensor fotoelétrico (normalmente, um diodo ou transistor sensível à luz) que detecta o pontinho de luz quando este passa em seu foco. Um sinal elétrico é então produzido, amplificado e transmitido para o computador. Este, por sua vez, é que controla o movimento de varredura e, por isso, consegue calcular as coordenadas do ponto, com base no tempo que o feixe de elétrons leva para partir do topo e chegar ao ponto de detecção.

As canetas ópticas variam segundo o tipo de sensor luminoso empregado, onde ele é colocado e onde se localizam os circuitos eletrônicos associados; ou, ainda, conforme permitam ou não diferentes modos operacionais. Até mesmo a maneira pela qual mostram que o sinal foi detectado serve para diferenciar os vários tipos de caneta óptica. Finalmente, como não poderia deixar de ser, elas são classificadas de acordo com o modelo de computador com o qual são compatíveis.

Todos esses fatores afetam o preço e o desempenho da caneta óptica, bem como suas características operacionais. Dentre estas, as principais são a *sensibilidade* e a *resolução*.

SENSIBILIDADE

A sensibilidade de uma caneta óptica revela a gama de intensidades luminosas que ela é capaz de detectar na tela. As melhores canetas captam todas ou quase todas as cores exibidas. As de qualidade inferior detectam apenas as cores mais brilhantes ou as intensidades mais fortes. Para muitas aplicações, isso será mais do que suficiente, embora imponha uma severa limitação na flexibilidade de uso, particularmente em aplicações gráficas, onde o fundo é normalmente escuro.

Um problema freqüente com as canetas ópticas é a chamada detecção espúria. Esta ocorre quando a sensibilidade é muito aumentada pela excessiva amplificação do sinal proveniente do sensor de luminosidade. Esse aumento amplia a probabilidade de a caneta óptica ser ativada por outras fontes de luz, tais como reflexos na tela ou a própria luz ambiente.

Para enfrentar esse problema, as canetas ópticas são equipadas com mecanismos destinados a impedir a ocorrência de detecções espúrias. Esses mecanismos podem ser constituídos, por exemplo, de células fotoelétricas e filtros de luz em proporções tais que se combinem com as características de cor do ponto na tela. Além disso, podem ser usados filtros eletrônicos para assegurar que a luz provenha realmente da tela (e não de uma fonte externa). Isso é possível porque os pontos na tela oscilam com uma freqüência característica (sessenta vezes por segundo). Assim, um circuito eletrônico simples (chamado "filtro passa-alto") é usado para selecionar apenas a luz que pisca com essa freqüência.

CANETAS DE ALTA RESOLUÇÃO

Uma vez que os problemas da sensibilidade e da detecção espúria tenham sido tecnicamente resolvidos, uma importante consideração adicional a fazer pode ser a resolução. Esta diz respeito à área mínima na tela que a caneta é capaz de detectar, que varia desde um único pixel na tela, nas melhores canetas ópticas, até um grupo inteiro de caracteres, nas piores.

A resolução de uma caneta depende do sensor que ela usa, da velocidade de reação e do método de colimação, ou de coleção da luz. Algumas canetas ópticas utilizam um tubo negro opaco para canalizar a luz até o sensor; outras empregam um sistema de lentes, ou pedaços curtos de fibra óptica. A alta resolução é importante se se quer desenhar com a pena, e não somente detectar posições. Uma caneta óptica com uma boa colimação conseguirá discriminar um ponto menor, e, conseqüentemente, será mais acurada.

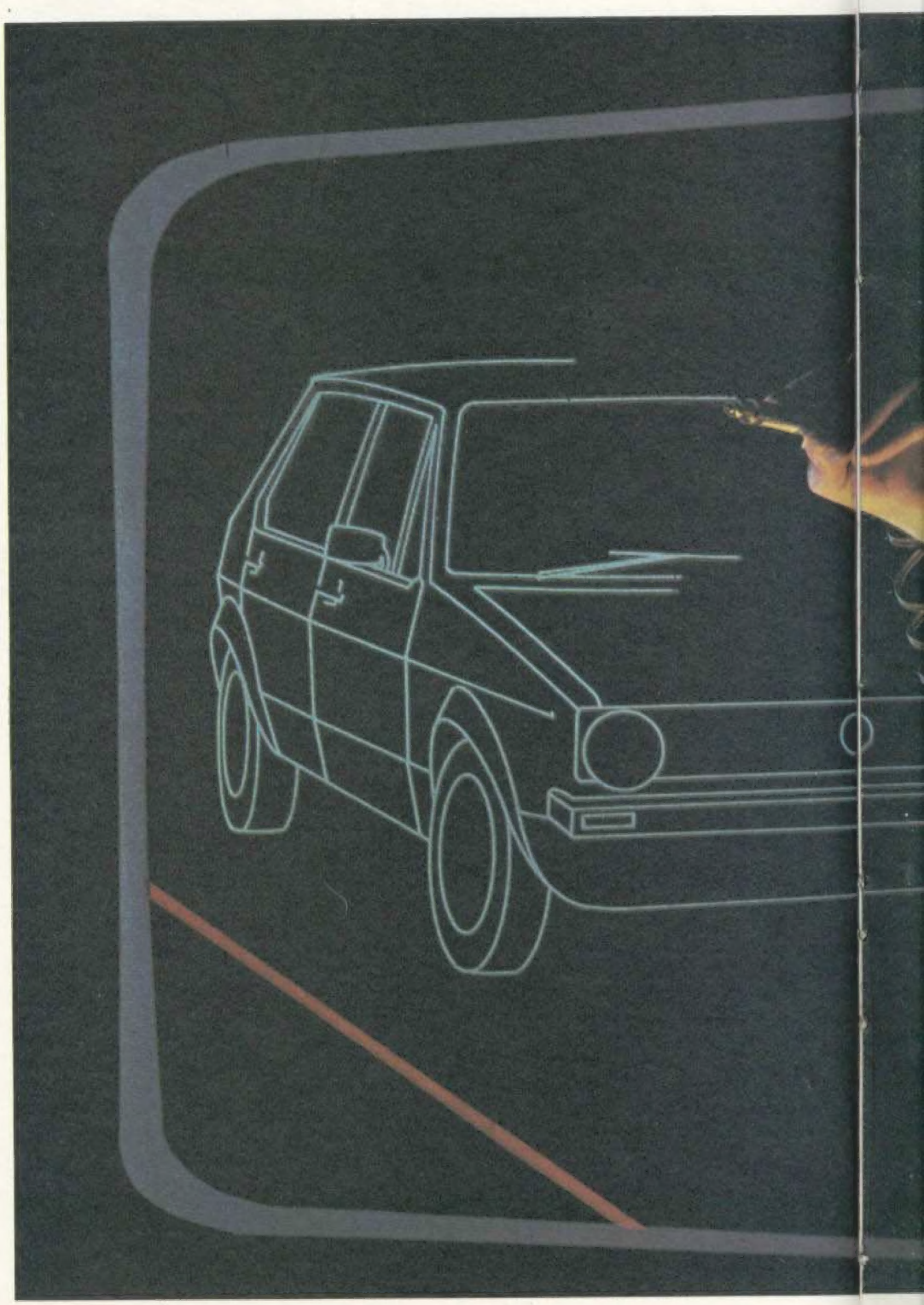
Os modelos de canetas ópticas diferem quanto à quantidade e qualidade da documentação e do software que as acompanham. Para funcionarem a contento, elas devem contar com um bom suporte por parte de rotinas de software. A programação é razoavelmente simples, qualquer que seja o tipo de caneta ou de computador ao qual ela está ligada. Alguns fabricantes adicionam uma fita cassete ou disco com alguns programas de exemplo, enquanto outros apenas fornecem listagens de rotinas.

COMO ESCOLHER UMA CANETA ÓPTICA

Há vários fatores a considerar na hora de se escolher uma caneta óptica. Por exemplo, a precisão da caneta. Embora algumas canetas tenham uma alta resolução nominal, isso não quer dizer que elas sejam capazes de conservar a precisão por muito tempo. Algumas canetas ópticas são tão imprecisas que uma reta traçada com elas sobre o vídeo aparecerá como uma série aleatória de pontos acesos e apagados.

Geralmente, o preço desse periférico é proporcional à qualidade. Alguns dos mais caros, por exemplo, têm dois dispositivos de que não falamos ainda: o primeiro serve para anunciar que a caneta detectou um ponto de luz válido na tela; o segundo controla o sinal a ser enviado ao computador.

Um diodo emissor de luz (LED) é um



equipamento usado em algumas canetas para indicar se houve detecção correta: ele não é essencial para desenhar, mas pode ser útil quando se usa a caneta para escolher opções na tela ou detectar posições.

O controle do sinal é normalmente exercido por um botão ou um interruptor, situado quase sempre no corpo da caneta, ao qual o usuário recorre quando quer enviar a informação ao computador. Ele pode ser acionado com duas finalidades: para reduzir a chance de detecções falsas ou espúrias, e para ter mais controle sobre que leituras devem ser feitas pela caneta.

Existem vários tipos de interruptores: nos modelos mais sofisticados, eles podem tomar a forma de um pequeno contato na ponta da caneta, de tal maneira que esta será ativada assim que for encostada na tela.

COMPATIBILIDADE

Antes de comprar uma caneta óptica, convém testar se ela funcionará com o seu tipo de computador. Como acontece com muitos outros periféricos, as canetas ópticas costumam funcionar apenas quando ligadas ao computador para o qual foram feitas.

Algumas vezes, você poderá resolver esse problema adquirindo uma interface especial para o seu computador.

Não se esqueça de que esse periférico precisa de software para funcionar: se você comprar uma caneta óptica que não sirva especificamente para o seu micro, provavelmente passará por maus momentos, tentando descobrir como ela funciona.

PROBLEMAS COM CANETAS ÓPTICAS

A limpeza é um item essencial para a manutenção das canetas ópticas. Como elas trabalham com luz em intensidades muito baixas, qualquer sujeira na ponta da caneta ou na tela tende a interferir no sinal luminoso detectado. Para evitar esse tipo de problema, limpe regularmente a caneta de seu micro com um produto de limpeza semelhante ao usado para gravadores.

Outro problema peculiar das canetas ópticas é que a sua proximidade tende a obscurecer uma parte da tela. Esse inconveniente, porém, só se tornará realmente grave se a tela estiver coberta de informações.

Antes de comprar uma caneta óptica, assegure-se de que nenhum outro periférico — como joysticks e tabletes digitalizadores — é mais apropriado para o que pretende fazer. Não se esqueça de que usar uma caneta óptica por longo tempo pode ser muito desconfortável, devido à posição vertical forçada em que ela deve ser mantida.

O mercado de micros tem se caracterizado por um forte crescimento no emprego de aplicações em que o usuário deve interagir com a tela. Essa tendência provocará certamente a disseminação das canetas ópticas. Com isso, seu preço tenderá a baixar, e os programas capazes de utilizá-las se multiplicarão. Para o usuário que está procurando um acessório barato e diferente para o seu micro, a caneta óptica pode ser, assim, a aquisição ideal.

ACELERE SEUS PROGRAMAS

Você prefere programar em BASIC, mas lamenta a morosidade do computador em executar certas tarefas? Não desanime. Como verá neste artigo, existem várias maneiras de acelerar seus programas.

A princípio, todos os computadores parecem extremamente rápidos, executando num piscar de olhos tarefas que consumiriam muitas horas de um ser humano. Mas, ao escrever um jogo de ação, um programa que contenha muitos cálculos ou até mesmo uma ordenação complicada usando o BASIC, o usuário começa a se decepcionar. Realmente a máquina demora para executar certas tarefas, e, depois de alguma experiência, é provável que você também acabe reclamando de sua lentidão...

Os programas mais rápidos são os escritos em código de máquina ou em linguagem Assembly, mas muitas pessoas optam pela facilidade da programação em BASIC. Um programa em BASIC, porém, nunca terá a mesma velocidade de execução de um programa em código de máquina, pois o computador gasta muito tempo traduzindo-o. O computador tem, já embutido como parte de seu hardware, um programa especial — o *interpretador* —, responsável pela tradução do BASIC para código de máquina.

Para os que não querem ou não gostam de escrever programas em código de máquina, mas ainda assim dão importância à velocidade e desejam extrair o máximo de rapidez do BASIC, temos algumas dicas. É importante, em primeiro lugar, tentar estruturar os programas adequadamente. Depois, deve-se selecionar os elementos do BASIC que o interpretador leva menos tempo para traduzir, de modo que cada linha de programa opere com velocidade máxima.

Cada máquina tem suas próprias limitações e, em conseqüência, cada programa requer um tratamento específico. Assim, não existem regras práticas que garantam a elaboração de programas perfeitos, mas, observando-se algumas recomendações, será possível torná-los bem mais eficientes.

CRONOMETRAGEM DO BASIC

Quase todas as linhas de microcomputadores possuem um temporizador interno que pode ser utilizado para comparar a velocidade de execução de programas. A rotina abaixo, que usaremos com exemplos dados mais adiante, permitirá que você observe as diferenças de velocidade existentes entre as várias formas de programação em BASIC. Os usuários do Apple precisarão recorrer a um cronômetro auxiliar de mão para fazer as comparações, pois os computadores dessa linha não dispõem de temporizador interno.

```
  1 POKE 23672,0: POKE 23673,0
  : POKE 23674,0
100 REM timer
120 FOR i=1 TO 100
130 GOSUB 200: NEXT i
140 LET b=PEEK 23672+256*PEEK
23673+65536*PEEK 23674
150 PRINT AT 5,5;(b-41)/5;"MIL
ISEGUNDOS"
160 STOP
200 REM
500 RETURN
```

```
100 TIME=0
110 FORK=1TO600:GOSUB200:NEXT
120 T=TIME+10
130 CLS:PRINTUSING" TEMPO USADO
 =##.### SEGUNDOS";T/60
140 END
200 REM
1000 RETURN
```

- VELOCIDADE DE EXECUÇÃO EM PROGRAMAS BASIC
- CRONOMETRAGEM DOS COMANDOS
- RELAÇÃO ENTRE ESTRUTURA, MEMÓRIA E VELOCIDADE
- ARMAZENAGEM DAS VARIÁVEIS
- FUNÇÕES MATEMÁTICAS
- MULTIPLICAÇÃO E DIVISÃO
- ORDENAÇÃO E BUSCA


```
10  REM  <USE UM CRONOMETRO
20  REM   PARA MEDIR O TEMPO>
100  HOME
110  PRINT  CHR$ (7)
120  PRINT : PRINT "INICIO"
130  FOR K = 1 TO 500
140  GOSUB 200
150  NEXT
160  PRINT  CHR$ (7)
170  PRINT : PRINT "FIM"
180  END
200  REM
1000  RETURN
```

```
100 TIMER=0
110 FOR K=1 TO 600:GOSUB 200:NE
XT
120 T=TIMER-170
130 CLS:PRINT" TEMPO USADO = ";
T/60;"SEGUNDOS"
140 END
200 REM
1000 RETURN
```

Em todos os programas, o comando **REM** da linha 200 será substituído, mais tarde, pelo nosso teste. No programa do Spectrum, esse comando aparece também na linha 100, devendo ali permanecer, já que faz parte da calibração do temporizador interno.

ESTRUTURA

Como a estruturação do programa desempenha um papel fundamental na velocidade de sua execução, convém recapitular aqui algumas regras básicas.

Todas as sub-rotinas usadas com mais freqüência devem ser posicionadas no início do programa, já que o interpretador procura cada linha requisitada por um **GOSUB** a partir do começo da listagem. Assim, quanto menor for o número da linha de uma sub-rotina, mais depressa ela será encontrada pelo interpretador. Você pode argumentar que um atraso de milisseguñdos não altera nada no programa. Mas a economia de alguns milissegundos aqui e ali acabará fazendo uma boa diferença.

Existem programas mal elaborados, que não passam de um emaranhado de comandos **GOTO** usados indiscriminadamente. Esse tipo de programa — às vezes chamado de "programa-espaguete" —, além de ser de difícil compreensão, perde muito na velocidade de execução. Ao contrário, quando a disposição das rotinas é bem planejada, a execução do programa torna-se bem mais rápida.

MEMÓRIA E VELOCIDADE

Comumente, os programas mais curtos são também os mais rápidos. Porém, os três objetivos do programador — velocidade, clareza e economia de memória — quase sempre estão em conflito. O uso de vários comandos numa só linha, por exemplo, economiza memória e acelera o programa, mas dificulta sua compreensão e eventuais correções.

Um programa que trabalha com velocidade máxima geralmente é mais longo e utiliza mais memória. A maioria das técnicas para se economizar memória traz prejuízo para a velocidade. É o caso do emprego de sub-rotinas.

DICAS PARA ACELERAR SEU BASIC

Nas matrizes, comece com o índice 0 e não com 1.

Defina as matrizes no início de cada programa.

Se possível, use variáveis inteiras em vez de variáveis de ponto flutuante.

Nos laços **FOR...NEXT**, não coloque uma variável depois do **NEXT**.

Use variáveis no lugar de números.

Dê nomes curtos às variáveis.

Coloque as sub-rotinas de uso mais freqüente no começo do programa.

Use laços **FOR...NEXT** no lugar de laços do tipo:

```
100 LET X=X+1 : IF X<20 THEN
GOTO 100
```

Numere as linhas com números mais baixos. Comece na linha 10 em vez de começar na linha 1000.

Use rotinas em código de máquina sempre que for possível. Torne-se um colecionador de sub-rotinas publicadas.

Nos programas voltados para cálculos, procure definir uma função que execute os cálculos repetitivos. Por exemplo:

```
10 DEF FN A(X)=X-(INT(X/360)*
360)
20 LET NUMERO=FN A(NUMERO)
```

é bem melnor que:

```
10 IF NUMERO>360 THEN LET NUMER
O=NUMERO-360: GOTO 10
```

Habitue-se a planejar seus programas. Além de erros, você evitará comandos **GOTO** e **GOSUB** desnecessários, que só comprometem a velocidade de execução.

Sub-rotinas economizam memória mas diminuem a rapidez do programa.

Remova todos os denominadores comuns. Por exemplo, troque:

```
10 LET X=Y/100+Z/100
```

por

```
10 LET X=(Y+Z)/100
```

Evite usar comandos **GOTO** sempre que possível.

Procure utilizar vários comandos na mesma linha. Perde-se em clareza, mas ganha-se em velocidade e economia.

Remova todos os espaços em branco e comandos **REM** desnecessários.

Ao usar um **IF...THEN**, coloque primeiro a condição que mais tende a ser FALSA.

Elas asseguram uma razoável economia de memória mas chamá-las toma tempo, o que não agrada aos programadores que buscam velocidade de execução. Como vimos anteriormente, o uso de sub-rotinas exige certo cuidado com a estruturação do programa. Sem isso, é impossível recuperar a velocidade que se perde ao chamá-las.

Uma linha como **LET A = VAL"100"** exemplifica bem a contradição que existe entre velocidade e economia de memória. Essa forma economiza memória, sem dúvida, mas, por outro lado, é de lenta execução se for comparada à forma mais comum **LET A = 100**.

VARIAVEIS

Saber como as variáveis são armazenadas na memória pode ser de grande utilidade para quem pretende acelerar a execução de programas. As variáveis são criadas à medida que aparecem no progama, sendo possível limpar a região em que se situam por meio dos comandos **RUN** ou **CLEAR**. Na maioria dos casos, uma nova variável provoca a ampliação dessa região para cima — o que não se aplica ao Sinclair Spectrum, que trabalha com as variáveis de uma maneira diferente.

Se criarmos, por exemplo, uma variável-cadeia e, depois, uma matriz numérica, todas as variáveis que se seguirem à matriz precisarão ser deslocadas para a parte superior da memória quando se acrescentar algo à cadeia.

Observe este programa:

```
100 LET T$=""
110 DIM A(1000)
120 LET T$=T$+"ACELERAR"
```

No BASIC da maioria dos micros, muito tempo de execução seria economizado se invertêssemos a ordem das duas primeiras linhas, ou seja, se a matriz fosse dimensionada antes de se definir a cadeia. Como está, o programa faz com que 5000 bytes sejam deslocados cada vez que se adiciona a palavra "ACELERAR" à variável-cadeia **T$**. Os usuários do Spectrum não precisam se preocupar, pois, como foi dito, esse micro trabalha com as variáveis de uma maneira diferente, não apresentando esse tipo de problema.

O uso de variável no lugar de números também assegura uma considerável economia de tempo. Essa regra aplica-se a todos os micros, menos ao Spectrum. No Apple, por exemplo, cada variável economiza cerca de cinco a dez milissegundos. Pode não parecer muito, mas imagine o que isso significa quan-

do se lida com um laço que é executado várias vezes. Já no Spectrum ocorre o oposto: a execução de um comando **LET C = 10 + 10** leva três milissegundos, enquanto a de **LET C = D + D** (onde D = 10) leva 4,2 milissegundos.

FUNÇÕES MATEMÁTICAS

Use a rotina apresentada anteriormente — que passaremos a chamar "cronômetro" — para testar estas duas formas de se obter a função potência.

```
200 LET C=4*4*4*4
```

ou, então:

```
200 LET C=4^4
```

É de se esperar que a função potência própria da máquina seja a mais rápida das duas. No Spectrum, por exemplo, a primeira forma leva seis milissegundos, e a segunda, 114 milissegundos. Estamos novamente diante de um caso em que a economia de memória (oferecida pela segunda forma) está em conflito com a velocidade.

Algumas vezes, lembramos que, no Spectrum, linhas do tipo

```
210 IF X>Y THEN LET Y=Y+1
220 IF X<Y THEN LET Y=Y-1
```

são executadas com maior velocidade quando substituídas por

```
10 LET Y = Y + (X>Y) - (X<Y)
```

O BASIC da linha Sinclair realmente permite esse tipo de comparação e programar assim é, sem dúvida, mais elegante. Mas, usando nossa rotina-cronômetro, podemos observar que a linha dupla é mais rápida que a simples.

Se você usa outro micro, teste o equivalente em sua máquina.

```
210 IF X>Y THEN Y=Y+1
220 IF X<Y THEN Y=Y-1
```

ou

```
210 Y=Y+(X<Y)-(X>Y)
```

Forneça, em cada caso, valores cabíveis para **X** e **Y**.

MULTIPLICAÇÃO E DIVISÃO

As expressões **C=D*0.5** e **C=D/2** executam exatamente o mesmo cálculo, mas, como mostram os testes, a multiplicação é mais rápida que a divisão.

Faça um teste com as sugestões apresentadas a seguir e tome nota dos resultados para referências futuras.

```
200 LET C=10+10
200 LET C=D+D  (ONDE D=10)
200 LET C=10*10
200 LET C=10/10
200 LET C=10+PI
200 LET C=SIN 10
200 LET C=COS 10
200 LET C=TAN 10
200 LET C=VAL "10"
200 LET C=10
200 LET C=D  (ONDE D=10)
200 PRINT AT 21,0;"TESTE"
200 PRINT AT 21,0;A$ (A$=TESTE)
200 PRINT AT 21,0; 10+1000+500+
 5.5
200 PRINT AT 21,0;D+E+F+G (ON-
DE D=10,E+1000,ETC)
```

```
200 C=10+10
200 C=D+D  (onde D=10)
200 C=10*10
200 C=10/10
200 C=10+PI
200 C=SIN(10)
200 C=COS(10)
200 C=TAN(10)
200 C=VAL("10")
200 C=10
200 C=D  (onde D=10)
200 PRINT"TESTE"
200 PRINT A$  (onde A$="TESTE")
200 PRINT 10+1000+500+5.5
200 PRINT D+E+F+G  (onde D=10,E
=1000,etc)
```

```
200 C=10+10
200 C=D+D  (ONDE D=10)
200 C=10*10
200 C=10/10
200 C=10+PI
200 C=SIN(10)
200 C=COS(10)
200 C=TAN(10)
200 C=VAL("10")
200 C=10
200 C=D  (ONDE D=10)
200 PRINT"TESTE"
200 PRINTA$  (ONDE A$="TESTE")
200 PRINT 10+1000+500+5.5
200 PRINT D+E+F+G  (ONDE D=10,E
=1000,ETC)
```

```
200 C=10+10
200 C=D+D  (ONDE D=10)
200 C=10*10
200 C=10/10
200 C=10+PI
200 C=SIN(10)
200 C=COS(10)
200 C=TAN(10)
200 C=VAL("10")
200 C=10
200 C=D  (ONDE D=10)
200 PRINT @260,"TESTE"
200 PRINT @260,A$ (A$="TESTE")
200 PRINT @260,10+1000+500+5.5
200 PRINT @260,D+E+F+G  (ONDE D
=10, E=1000, ETC)
```

ORDENAÇÃO E BUSCA

Como as técnicas de ordenação já foram tema de artigos anteriores, não entraremos aqui em maiores detalhes. Como vimos, a ordenação tipo *Shell-Metzner* é a mais indicada quando se trata de manipular um número de itens superior a cem. Esse tipo de ordenação tem a característica de ser tanto mais rápido quanto maior for o número de itens. Porém, como era de se esperar, também nesse caso a velocidade envolve certo sacrifício de memória.

Ao contrário do que muitos pensam, ordenação e busca não são a mesma coisa. Buscar é acessar um dado (ou um conjunto de dados) o mais rapidamente possível; ordenar é colocar vários dados em ordem. Suponhamos que você precise encontrar o telefone de certa pessoa em uma lista armazenada no computador. Ora, não haveria vantagem nenhuma se o tempo gasto pelo computador para encontrar tal número fosse maior que o tempo necessário para achá-lo numa lista telefônica. Por isso, quando se trata de ordenar ou buscar, sempre se procura o método mais rápido.

A busca serial (listada a seguir) simula o procedimento de uma pessoa que procura determinado dado analisando item por item de uma lista.

```
10 DATA "ANTILOPE","CACHORRO"
,"ELEFANTE","GOLFINHO","JUMEN
TO","LEOPARDO","MACACO","PERI
QUITO","RAPOSA","SAPO"
20 RESTORE 10: DIM B$(10,9)
30 CLS : FOR i=1 TO 10: READ
B$(i): PRINT AT i,5;B$(i):
NEXT i
40 POKE 23658,8: INPUT "Qual
o animal ?";A$
50 FOR X=1 TO 10
60 IF B$(X, TO LEN A$)=A$
THEN  PRINT  FLASH 0;AT X,13;
"": FLASH 1;" ACHEI": GOTO 40
70 NEXT X
```

```
10 REM <busca serial>
20 DIM B$(10)
30 CLS:FORI=1TO10:READB$(I):LOC
ATE0,I+3:PRINTB$(I):NEXT
40 LOCATE0,17:INPUT"Qual a anim
al";A$
50 FORX=1TO10
60 IF B$(X)=A$ THEN LOCATE13,X+
3:PRINT"<< ACHEI":X=10
70 NEXT:GOTO40
```

```
80 DATA ANTILOPE,CACHORRO,ELEFA
NTE,GOLFINHO,JUMENTO,LEOPARDO,M
ACACO,PERIQUITO,RAPOSA,SAPO
```

```
10  REM  <BUSCA SERIAL>
20  DIM B$(10)
30  HOME : FOR I = 1 TO 10
35  READ B$(I): PRINT  TAB( 11)
;B$(I)
40  NEXT : PRINT : PRINT
45  INPUT "QUAL O ANIMAL ?";A$
50  FOR X = 1 TO 10
60  IF B$(X) = A$ THEN  VTAB (X
): PRINT "ACHEI >>":X = 10
70  NEXT : VTAB (13): GOTO 45
80  DATA  ANTILOPE,CACHORRO,ELE
FANTE,GOLFINHO,JUMENTO,LEOPARDO
,MACACO,PERIQUITO,RAPOSA,SAPO
```

```
10 REM BUSCA SERIAL
20 DIM B$(10)
30 CLS:FOR I=1 TO 10:READ B$(I)
:PRINT @33+I*32,B$(I);:NEXT
40 PRINT @417,STRING$(30,32);ST
RING$(30,8);:INPUT"QUAL O ANIMA
L ";A$
50 FOR X=1 TO 10
60 IF B$(X)=A$ THEN PRINT @46+X
*32,CHR$(191);"ACHEI";:X=10
70 NEXT:GOTO 40
80 DATA ANTILOPE,CACHORRO,ELEFA
NTE,GOLFINHO,JUMENTO,LEOPARDO,M
ACACO,PERIQUITO,RAPOSA,SAPO
```

A rotina de busca binária (listada a seguir), um pouco mais difícil de se programar que a de busca serial, é bem mais rápida que esta. A diferença de velocidade entre ambas torna-se maior à medida que a quantidade de dados da lista aumenta.

A busca binária é mais rápida que a serial porque não utiliza o processo de inspeção item por item. Ela requer que todos os dados tenham sido previamente colocados em ordem alfabética ou numérica. Cumprido esse pré-requisito, o computador analisa o elemento central da lista, para definir se se orientará para sua metade superior ou para sua metade inferior.

A metade escolhida será novamente dividida ao meio e, a partir da análise desse novo elemento central, outra metade será escolhida e igualmente dividida ao meio.

O processo se repete até que o item desejado seja encontrado, ou, caso contrário, até que não haja mais nenhuma possibilidade de dividir a lista ao meio.

No início, o computador não procura por itens que sejam totalmente equivalentes, mas apenas compara a primeira letra, verificando se, na ordem alfabética, ela vem antes ou depois da do item desejado.

```
10 CLS : RESTORE
20 LET t=10: LET b=1
30 DIM n$(10,10)
40 FOR c=1 TO 10
50 READ n$(c)
60 NEXT c
70 INPUT "NOME DO ANIMAL ";a$
75 LET a$=a$+"          "( TO
10-LEN a$)
80 PAUSE 50
95 CLS
100 IF n$(t)=a$ THEN  PRINT n$
(t),t: GOTO 200
110 IF n$(B)=a$ THEN  PRINT n$
(b),b: GOTO 200
120 LET p=INT (0.5+(t+b)/2)
130 IF n$(p)=a$ THEN  PRINT n$
(p),p: GOTO 200
140 IF n$(p)>a$ THEN  LET t=p
150 IF n$(p)<a$ THEN  LET b=p
160 IF t-b=1 THEN  PRINT "   N
AO ACHEI": GOTO 200
170 GOTO 100
200 IF INKEY$="" THEN  GOTO
 200
210 RUN
580 DATA "Antilope","Cachorro"
,"Elefante","Golfinho","Jument
o","Leopardo","Macaco","Periqu
ito","Raposa","Sapo"
```

```
10 CLS
20 T=10:B=1
30 DIM N$(10)
40 FOR C=1 TO 10
50 READ N$(C)
60 NEXT
70 INPUT"nome do animal";A$
80 TIME=0
90 IF TIME<50 THEN 90 ELSE CLS
100 IF N$(T)=A$ THEN PRINTN$(T)
,T:GOTO200
110 IF N$(B)=A$ THEN PRINTN$(B)
,B:GOTO200
120 P=INT(.5+(T+B)/2)
130 IF N$(P)=A$ THEN PRINTN$(P)
,P:GOTO200
140 IF N$(P)>A$ THEN T=P
150 IF N$(P)<A$ THEN B=P
160 IF T-B=1 THEN PRINT"    não
 achei":GOTO200
170 GOTO 100
200 IF INKEY$="" THEN 200
210 RUN
580 DATA ANTILOPE,CACHORRO,ELEF
ANTE,GOLFINHO,JUMENTO,LEOPARDO,
MACACO,PERIQUITO,RAPOSA,SAPO
```

```
10  HOME
20 T = 10:B = 1
30  DIM N$(10)
40  FOR C = 1 TO 10
50  READ N$(C)
60  NEXT
70  INPUT "QUAL O ANIMAL ?";A$
```

```
80  FOR TT = 0 TO 200: NEXT
100  IF N$(T) = A$ THEN  PRINT
N$(T),T: GOTO 200
110  IF N$(B) = A$ THEN  PRINT
N$(B),B: GOTO 200
120 P =  INT (.5 + (T + B) / 2)

130  IF N$(P) = A$ THEN  PRINT
N$(P),P: GOTO 200
140  IF N$(P) > A$ THEN T = P
150  IF N$(P) < A$ THEN B = P
160  IF T - B = 1 THEN  PRINT "
    NAO ACHEI": GOTO 200
170  GOTO 100
200  GET Z$: IF Z$ = "" THEN 20
0
210  RUN
580  DATA   ANTILOPE,CACHORRO,E
```

```
LEFANTE,GOLFINHO,JUMENTO,LEOPAR
DO,MACACO,PERIQUITO,RAPOSA,SAPO
```

```
10 CLS
20 T=10:B=1
30 DIM N$(10)
40 FOR C=1 TO 10
50 READ N$(C)
60 NEXT
70 INPUT" NOME DO ANIMAL ";A$
80 TIMER=0
90 IF TIMER<50 THEN 90 ELSE CLS
100 IF N$(T)=A$ THEN PRINT N$(T
),T:GOTO 200
110 IF N$(B)=A$ THEN PRINT N$(B
),B:GOTO 200
120 P=INT(.5+(T+B)/2)
130 IF N$(P)=A$ THEN PRINT N$(P
),P:GOTO 200
140 IF N$(P)>A$ THEN T=P
150 IF N$(P)<A$ THEN B=P
160 IF T-B=1 THEN PRINT"    NAO
 ACHEI":GOTO 200
170 GOTO 100
200 IF INKEY$="" THEN 200
210 RUN
580 DATA ANTILOPE,CACHORRO,ELEF
ANTE,GOLFINHO,JUMENTO,LEOOPARDO
,MACACO,PERIQUITO,RAPOSA,SAPO
```

Como foi dito, não existem regras práticas para a obtenção de maior velocidade de execução em nossos programas. Os melhores resultados são conseguidos dedicando-se atenção especial a detalhes que, isolados, parecem ser insignificantes, mas que, juntos, fazem a diferença. Alguns desses detalhes são lembrados no quadro da página 932. Observando as recomendações ali contidas, você certamente obterá alguma economia de tempo. Mas não se esqueça de que, muitas vezes, a rapidez de um programa procede de uma descoberta individual, resultante, até mesmo, de um simples acaso.

UM ASSISTENTE PARA O DOS

Se você já dispõe de um acionador de disquetes mas tem preguiça de aprender os comandos do sistema operacional (DOS), utilize um programa assistente e durma tranqüilo.

Qualquer que seja a aplicação a que se destine o seu microcomputador, convém adquirir um acionador de disquetes. Desse modo, você poderá dispor de um sistema bem mais confiável e rápido do que as fitas cassete.

FUNÇÕES DO DOS

Do ponto de vista operacional, contudo, esse sistema apresenta uma pequena desvantagem. Com ele, o usuário é obrigado a aprender um número adicional de comandos, específicos para operações com disquetes. Tais comandos fazem parte do chamado Sistema Operacional de Discos (DOS, sigla decorrente de seu nome em inglês). Eles funcionam mais ou menos da mesma maneira nos diferentes modelos de computador, variando apenas a forma com que são escritos, e a sua sintaxe de uso.

As funções básicas de um DOS são:
- examinar o conteúdo de um disquete (catálogo);
- copiar um arquivo de um disquete para outro ou fazer uma cópia com nome diferente no mesmo disquete;
- mudar o nome de um arquivo;
- apagar um arquivo do disquete;
- listar um arquivo na tela do computador ou na impressora;
- examinar qual o espaço que está disponível no disquete;
- proteger ou desproteger um arquivo contra apagamento ou modificação;
- executar programas em BASIC ou em linguagem de máquina etc.

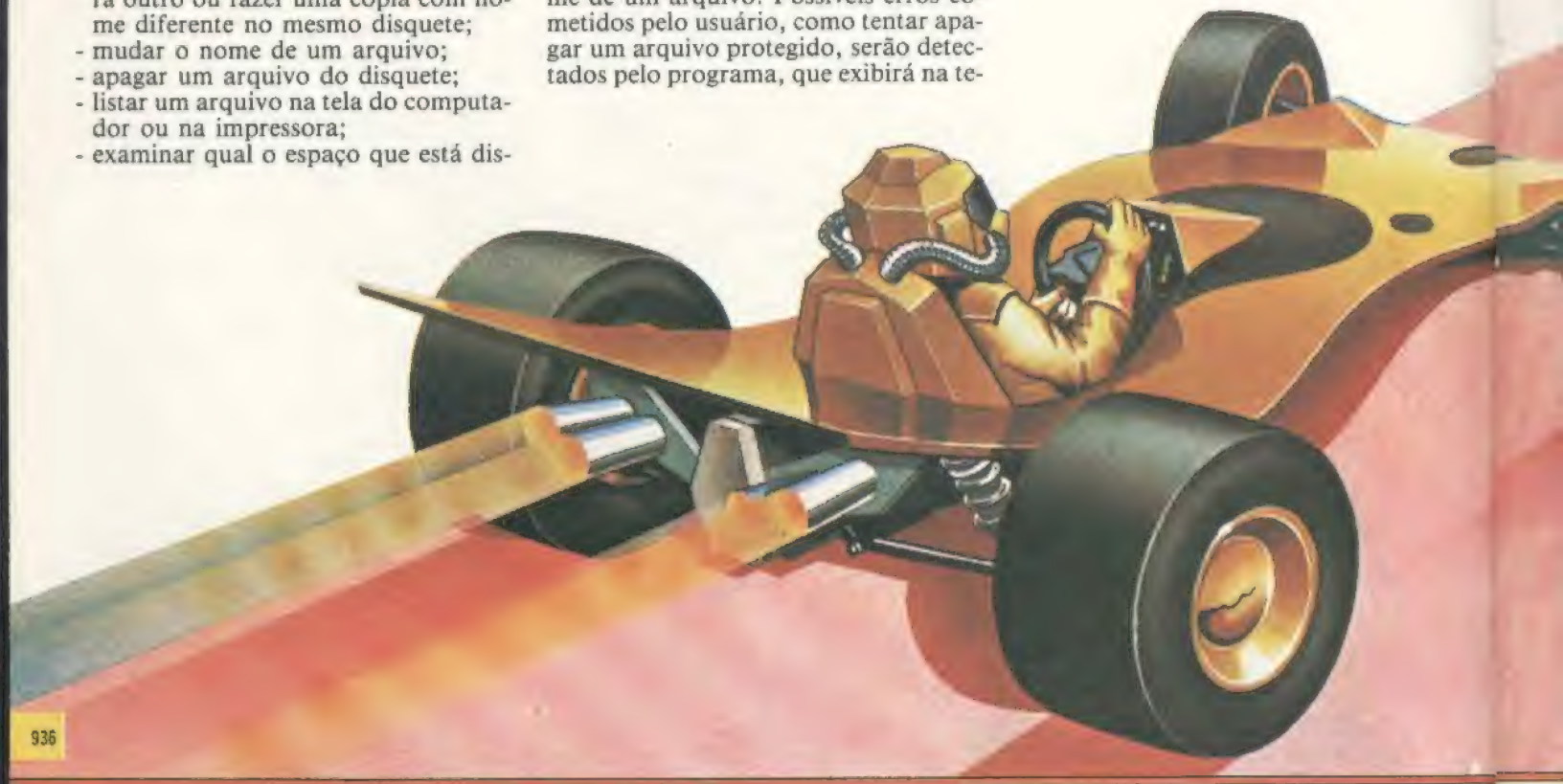
Nem sempre, contudo, o usuário está disposto a aprender em detalhe todas essas funções. De certo modo, estas caracterizam uma "linguagem" de comandos, que em alguns computadores é parte das instruções do BASIC, enquanto em outros é um sistema totalmente separado, do qual o interpretador BASIC é apenas um programa a mais.

A listagem apresentada aqui serve como "assistente" para a operação dos vários comandos do DOS; ela dispensa a necessidade de aprendê-los em detalhe. Seu funcionamento se apóia em um "cardápio" de opções, mostrado na tela assim que é executado. Escolhendo uma das opções apresentadas, o programa automaticamente seleciona e executa o comando correspondente do DOS. Em muitos casos, será necessário entrar informações adicionais, tal como o nome de um arquivo. Possíveis erros cometidos pelo usuário, como tentar apagar um arquivo protegido, serão detectados pelo programa, que exibirá na tela a mensagem correspondente.

Neste artigo abordaremos versões para os computadores das linhas TRS-80, TRS-Color, MSX, Apple II e TK-2000. Os micros da linha Sinclair também podem ser conectados a disquetes, ou "microdrives", fitas em alça sem fim, de comportamento semelhante aos disquetes; mas como estes periféricos não são encontrados no país, deixamos de apresentar uma versão para eles.

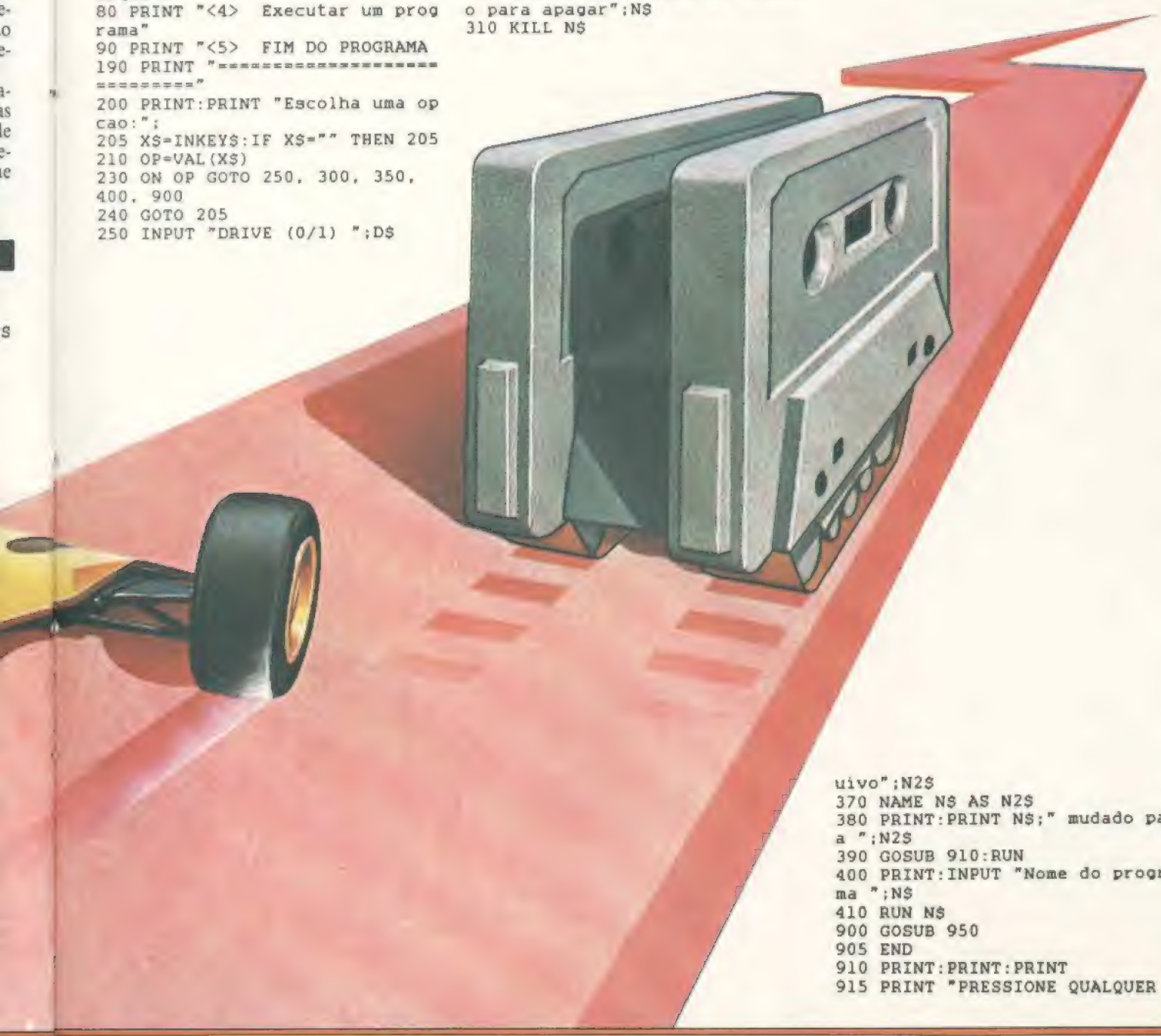
Digite os programas a seguir e armazene-os em um disquete. Para testar as opções do menu sem correr o risco de perder algo importante, copie no disquete alguns arquivos e programas de que não vai precisar.

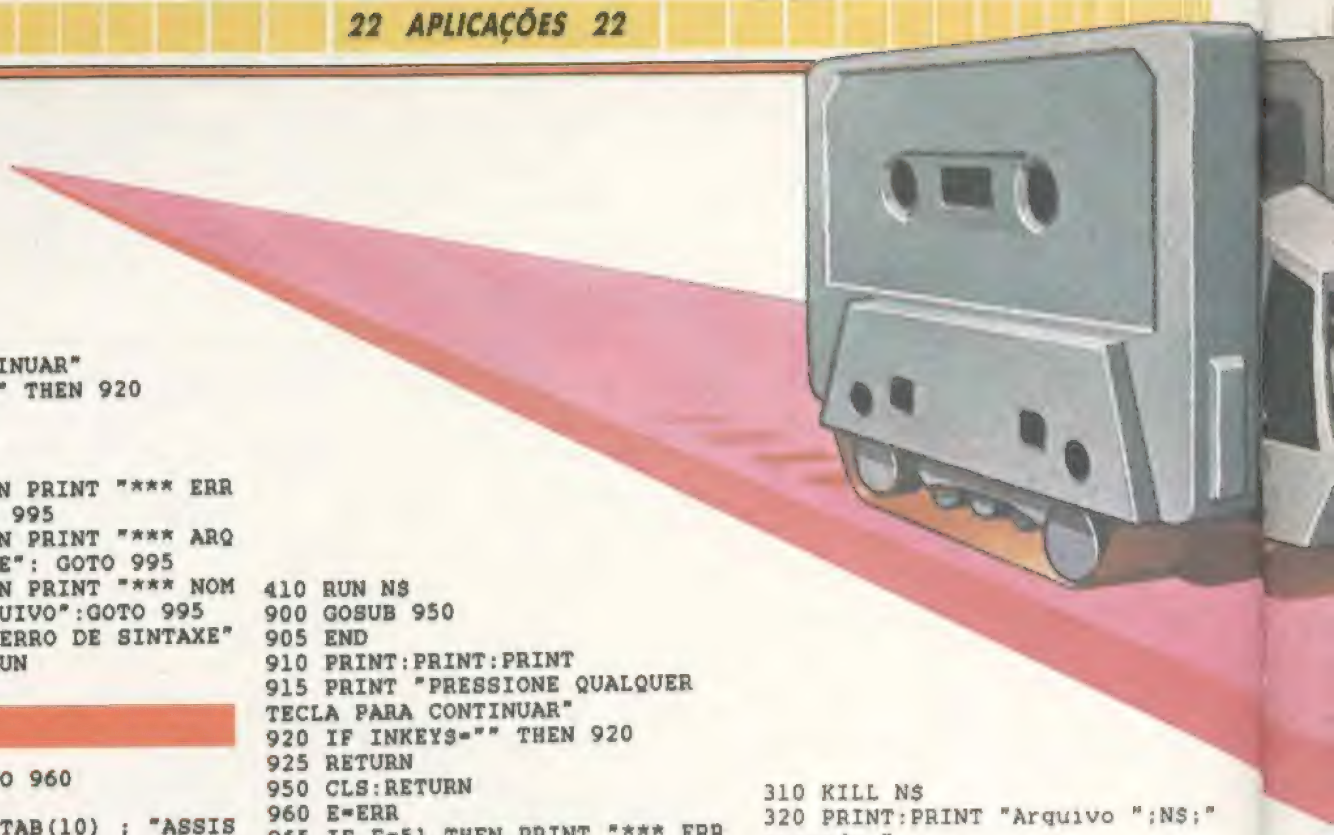
```
10 ON ERROR GOTO 960
20 GOSUB 950
30 PRINT:PRINT TAB(10) ; "ASSIS
TENTE D.O.S."
35 PRINT
```



- O QUE É UM SISTEMA OPERACIONAL DE DISCOS
- FUNÇÕES PRINCIPAIS DO DOS
- UM ASSISTENTE INTELIGENTE
- INFORMAÇÕES ADICIONAIS
- DETECÇÃO DE ERROS
- COMO FUNCIONA O PROGRAMA
- EXIBIÇÃO DO MENU
- LIMITAÇÕES

```
40 PRINT "=====================
=========="
50 PRINT "<1>  Listar conteudo
do disco"
60 PRINT "<2>  Apagar um arquiv
o do disco"
70 PRINT "<3>  Mudar nome de um
arquivo"
80 PRINT "<4>  Executar um prog
rama"
90 PRINT "<5>  FIM DO PROGRAMA
190 PRINT "=====================
=========="
200 PRINT:PRINT "Escolha uma op
cao:";
205 X$=INKEY$:IF X$="" THEN 205
210 OP=VAL(X$)
230 ON OP GOTO 250, 300, 350,
400, 900
240 GOTO 205
250 INPUT "DRIVE (0/1) ";D$
251 IF D$<>"0" AND D$<>"1" THEN
GOTO 250
252 GOSUB 950
255 PRINT TAB(10);"C A T A L O
G O"
260 PRINT:CMD "D:"+D$
270 GOSUB 910:RUN
300 PRINT:INPUT "Nome do arquiv
o para apagar";N$
310 KILL N$
320 PRINT:PRINT "Arquivo ";N$;"
apagado."
330 GOSUB 910:RUN
350 PRINT:INPUT "Nome do arquiv
o a ser mudado";N$
360 INPUT "Novo nome para o arq
uivo";N2$
370 NAME N$ AS N2$
380 PRINT:PRINT N$;" mudado par
a ";N2$
390 GOSUB 910:RUN
400 PRINT:INPUT "Nome do progra
ma ";N$
410 RUN N$
900 GOSUB 950
905 END
910 PRINT:PRINT:PRINT
915 PRINT "PRESSIONE QUALQUER
```





```
TECLA PARA CONTINUAR"
920 IF INKEY$="" THEN 920
925 RETURN
950 CLS:RETURN
960 E=ERR
965 IF E=51 THEN PRINT "*** ERR
O INTERNO":GOTO 995
970 IF E=53 THEN PRINT "*** ARQ
UIVO INEXISTENTE": GOTO 995
990 IF E=56 THEN PRINT "*** NOM
E ILEGAL DE ARQUIVO":GOTO 995
992 PRINT "*** ERRO DE SINTAXE"
995 GOSUB 910:RUN
```

T

```
10 ON ERROR GOTO 960
20 GOSUB 950
30 PRINT:PRINT TAB(10) ; "ASSIS
TENTE D.O.S."
35 PRINT
40 PRINT "----------------------
----------"
50 PRINT "<1>  LISTAR CONTEUDO
DO DISCO"
60 PRINT "<2>  APAGAR UM ARQUIV
O DO DISCO"
70 PRINT "<3>  MUDAR NOME DE UM
ARQUIVO"
80 PRINT "<4>  EXECUTAR UM PROG
RAMA
90 PRINT "<5>  FIM DO PROGRAMA
190 PRINT "----------------------
----------"
200 PRINT:PRINT "ESCOLHA UMA OP
CAO:";
205 X$=INKEY$:IF X$="" THEN 205
210 OP=VAL(X$)
230 ON OP GOTO 250, 300, 350,
400, 900
240 GOTO 205
250 GOSUB 950
255 PRINT TAB(10);"C A T A L O
G O"
260 PRINT:DIR
270 GOSUB 910:RUN
300 PRINT:INPUT "NOME DO ARQUIV
O PARA APAGAR";N$
310 KILL N$
320 PRINT:PRINT "ARQUIVO ";N$;"
APAGADO."
330 GOSUB 910:RUN
350 PRINT:INPUT "NOME DO ARQUIV
O A SER MUDADO";N$
360 INPUT "NOVO NOME PARA O ARQ
UIVO";N2$
370 NAME N$ AS N2$
380 PRINT:PRINT N$;" MUDADO PAR
A ";N2$
390 GOSUB 910:RUN
400 PRINT:INPUT "NOME DO PROGRA
MA ";N$
410 RUN N$
900 GOSUB 950
905 END
910 PRINT:PRINT:PRINT
915 PRINT "PRESSIONE QUALQUER
TECLA PARA CONTINUAR"
920 IF INKEY$="" THEN 920
925 RETURN
950 CLS:RETURN
960 E=ERR
965 IF E=51 THEN PRINT "*** ERR
O INTERNO":GOTO 995
970 IF E=53 THEN PRINT "*** ARQ
UIVO INEXISTENTE": GOTO 995
990 IF E=56 THEN PRINT "*** NOM
E ILEGAL DE ARQUIVO":GOTO 995
992 PRINT "*** ERRO DE SINTAXE"
995 GOSUB 910:RUN
```

M

```
10 ON ERROR GOTO 960
20 GOSUB 950
30 PRINT:PRINT TAB(10) ; "ASSIS
TENTE D.O.S."
35 PRINT
40 PRINT "----------------------
----------"
50 PRINT "<1>  Listar conteúdo
do disco"
60 PRINT "<2>  Apagar um arquiv
o do disco"
70 PRINT "<3>  Mudar nome de um
arquivo"
80 PRINT "<4>  Executar um prog
rama"
90 PRINT "<5>  FIM DO PROGRAMA
190 PRINT "----------------------
----------"
200 PRINT:PRINT "Escolha uma op
ção:";
205 X$=INKEY$:IF X$="" THEN 205
210 OP=VAL(X$)
230 ON OP GOTO 250, 300, 350,
400, 900
240 GOTO 205
250 GOSUB 950
255 PRINT TAB(10);"C A T-A L O
G O"
260 PRINT:FILES
270 GOSUB 910:RUN
300 PRINT:INPUT "Nome do arquiv
o para apagar";N$
310 KILL N$
320 PRINT:PRINT "Arquivo ";N$;"
apagado."
330 GOSUB 910:RUN
350 PRINT:INPUT "Nome do arquiv
o a ser mudado";N$
360 INPUT "Novo nome para o arq
uivo";N2$
370 NAME N$ AS N2$
380 PRINT:PRINT N$;" mudado par
a ";N2$
390 GOSUB 910:RUN
400 PRINT:INPUT "Nome do progra
ma ";N$
410 RUN N$
900 GOSUB 950
905 END
910 PRINT:PRINT:PRINT
915 PRINT "PRESSIONE QUALQUER
TECLA PARA CONTINUAR"
920 IF INKEY$="" THEN 920
925 RETURN
950 CLS:RETURN
960 E=ERR
965 IF E=51 THEN PRINT "*** ERR
O INTERNO":GOTO 995
970 IF E=53 THEN PRINT "*** ARQ
UIVO INEXISTENTE": GOTO 995
990 IF E=56 THEN PRINT "*** NOM
E ILEGAL DE ARQUIVO":GOTO 995
992 PRINT "*** ERRO DE SINTAXE"
995 GOSUB 910:RUN
```

CP

```
10 ON ERR GOTO 960
15 LET D$=CHR$(4)
20 GOSUB 950
30 PRINT:PRINT TAB(10) ; "ASSIS
TENTE D.O.S."
35 PRINT
40 PRINT "----------------------
----------"
50 PRINT "<1>  LISTAR CONTEUDO
DO DISCO"
```

```
60 PRINT "<2>  APAGAR UM ARQUIV
O DO DISCO"
65 PRINT "<3>  MUDAR NOME DE UM
ARQUIVO"
70 PRINT "<4>  EXECUTAR UM PROG
RAMA"
75 PRINT "<5>  PROTEGER UM ARQU
IVO"
80 PRINT "<6>  DESPROTEGER UM A
RQUIVO"
85 PRINT "<7>  VERIFICAR UM ARQ
UIVO"
90 PRINT "<8>  FIM DO PROGRAMA
190 PRINT "==================
=========="
200 PRINT:PRINT "ESCOLHA UMA OP
CAO:";
205 GET X$
210 LET OP=VAL(X$)
230 ON OP GOTO 250, 300, 350,
400, 450, 500, 550, 900
240 GOTO 205
250 GOSUB 950
255 PRINT TAB(10);"C A T A L O
G O"
260 PRINT:PRINT D$;"CATALOG"
270 GOSUB 910:RUN
300 PRINT:INPUT "NOME DO ARQUIV
O PARA APAGAR";N$
310 PRINT D$;"DELETE ";N$
320 PRINT:PRINT "ARQUIVO ";N$;"
APAGADO."
330 GOSUB 910:RUN
350 PRINT:INPUT "NOME DO ARQUIV
O A SER MUDADE";N$
360 INPUT "NOVO NOME PARA O ARQ
UIVO";N2$
370 PRINT D$;"RENAME
";N$;",";N2$
380 PRINT:PRINT N$;" MUDADO PAR
A ";N2$
390 GOSUB 910:RUN
400 PRINT:INPUT "NOME DO PROGRA
MA ";N$
410 PRINT D$;"RUN ";N$
450 PRINT:INPUT "NOME DO ARQUIV
O PARA PROTEGER ";N$
460 PRINT D$;"LOCK ";N$
470 PRINT:PRINT "ARQUIVO ";N$;"
PROTEGIDO."
480 GOSUB 910:RUN
500 PRINT:INPUT "NOME DO ARQUIV
O PARA DESPROTEGER ";N$
510 PRINT D$;"UNLOCK ";N$
520 PRINT:PRINT "ARQUIVO ";N$;"
DESPROTEGIDO."
530 GOSUB 910:RUN
550 PRINT:INPUT "NOME DO ARQUIV
O PARA VERIFICAR ";N$
560 PRINT D$;"VERIFY ";N$
570 PRINT:PRINT "ARQUIVO ";N$;"
CORRETO."
580 GOSUB 910:RUN
900 GOSUB 950
905 END
910 PRINT:PRINT:PRINT
915 PRINT "PRESSIONE RETURN
PARA CONTINUAR"
```

```
920 GET X$
925 RETURN
950 HOME:RETURN
960 LET E=PEEK (222)
965 IF E=8 THEN PRINT "*** ERRO
DE E/S":GOTO 995
970 IF E=6 THEN PRINT "*** ARQU
VO INEXISTENTE": GOTO 995
970 IF E=10 THEN PRINT "*** ARQ
UIVO PROTEGIDO": GOTO 995
990 IF E=13 THEN PRINT "*** TIP
O ILEGAL DE ARQUIVO":GOTO 995
992 IF E=11 THEN PRINT "*** ERR
O DE SINTAXE"
995 GOSUB 910:RUN
```

COMO FUNCIONA

As linhas 10 a 240 mostram o menu na tela. Ele varia conforme o modelo do computador, pois nem todas as suas funções são iguais, ou acessíveis por meio de um programa.

A linha 230 (**ON...GOTO**) desvia o programa para o trecho correspondente a cada função. O comando **ON ERR GOTO** logo na primeira linha serve para detectar erros de processamento de um comando do DOS. Caso isso aconteça, ele desvia o programa para o trecho em que será mostrada uma mensagem de erro (linha 960 em diante).

Após acionar a função do DOS que foi solicitada, o programa se auto-executa novamente, mostrando a tela de opções. A única exceção, em todos os microcomputadores, acontece quando se solicita a execução de um outro programa: não há retorno automático ao menu inicial, e você deve executá-lo mais uma vez, se quiser.

As linhas 910 e 950 marcam o início de duas sub-rotinas usadas repetidamente ao longo do programa. A primeira (910) serve para colocar uma mensagem no vídeo, avisando que o usuário deve pressionar a tecla <**RETURN**> ou <**ENTER**> para limpar a tela e voltar ao menu principal (nos micros das linhas MSX, TRS-80 e TRS-Color, pode-se pressionar qualquer tecla).

A sub-rotina na linha 950, por sua vez, é usada apenas para limpar a tela; ela foi colocada aí para facilitar a conversão do programa para outros micros. Assim, em vez de ter de alterar todas as linhas do programa onde ocorre um **CLS**, ou um **HOME** (e são muitas), é mais fácil modificar apenas uma.

Os micros da linha TRS-Color não têm um sistema operacional separado, como é o caso dos modelos das linhas TRS-80 e MSX. Ao se inserir o conector do acionador de disquetes na porta de cartuchos da máquina, o BASIC armazenado na ROM do computador fica apto para usar os comandos exclusivos para a operação do disquete. Esses comandos são poucos, em contraste, mais uma vez, com o TRS-80. As instruções principais — de apagamento (**KILL**), mudança de nome (**NAME**) e execução de programas em BASIC, diretamente do disco (**RUN**) —, contudo, são idênticas às do TRS-80; por isso os programas para ambas as linhas são similares. A única diferença diz respeito ao comando usado para listar o catálogo do disquete (**DIR**).

Os micros da linha TRS-80 contam com um sistema operacional separado, que pode ser invocado fora do BASIC. Esse sistema é armazenado nas trilhas do meio do disquete, e é carregado automaticamente toda vez que o computador é inicializado. Entretanto, o BASIC de disco tem alguns comandos semelhantes, em operação, aos comandos do DOS. Embora em pequeno número, esses comandos são a base do funcionamento do programa aqui apresentado. Os principais são os de apagamento (**KILL**), mudança de nome (**NAME**) e execução de programas em BASIC, diretamente do disco (**RUN**). Eles são idênticos aos do TRS-Color; por isso os programas para ambas as linhas são similares. A única diferença diz respeito à instrução usada para listar o catálogo do disquete (**DIR**) no TRS-Color. No TRS-80 é usado o comando genérico **CMD**, que dá acesso direto, a partir do BASIC, a várias funções do DOS. Para obter o catálogo, por exemplo, usa-se **CMD "D:0"** ou **CMD "D:1"**, dependendo do acionador em que está o disquete. Note que o número do acionador é perguntado pelo programa.

Como no caso anterior, os micros da linha MSX têm um sistema operacional separado, que pode ser invocado fora do BASIC. Armazenado nas trilhas iniciais do disquete, esse sistema é carregado automaticamente toda vez que o computador é inicializado. Entretanto, o BASIC de disco conta com algumas instruções semelhantes, em operação, aos comandos do DOS. Essas instruções formam a base do funcionamento do programa aqui apresentado. Os comandos principais são os de apagamento (**KILL**), mudança de nome (**NAME**) e execução de programas em BASIC, diretamente do disco (**RUN**). Eles são idênticos aos do TRS-80 e do TRS-Color; por isso os programas para ambas as linhas são similares. A única diferença refere-se ao comando que lista o catálogo do disquete (**DIR** no TRS-Color, **CMD** no TRS-80). No MSX, esse comando se chama **FILES**, e, em operação, é equivalente à instrução **DIR** do seu sistema operacional. Os acionadores de disquetes podem ser chamados de A ou B. Note que o programa pergunta o nome do acionador. O programa utiliza todos os comandos referidos.

Assim como os microcomputadores da linha TRS-Color, os modelos das linhas Apple e TK-2000 não dispõem de um sistema operacional de discos separado do BASIC. Ao ser utilizado com disquete, o interpretador de ambas as máquinas passa automaticamente a ter acesso aos comandos específicos necessários à sua operação.

Entretanto, há um pequeno problema: os comandos básicos, como **RUN, DELETE, RENAME, LOCK, UNLOCK, VERIFY** e **CATALOG**, só respondem quando usados de modo direto (isto é, fora de um programa). Para executá-los a partir de um programa, é necessário colocar a cadeia de comandos dentro de um **PRINT**, o qual deve transmitir antes o caractere ASCII 4; só agindo assim a manobra surtirá efeito.

Isso é explorado pelo programa, que, aliás, é o mais extenso de todos, em virtude da maior flexibilidade do Apple para explorar comandos do DOS a partir de um programa em BASIC.

LIMITAÇÕES

O programa apresentado atinge seu principal objetivo, pois, de fato, permite ao usuário operar com disquetes sem conhecer os comandos adicionais específicos. Porém, como todo programa que tenta substituir uma linguagem de comandos por um sistema de escolha por cardápios, o nosso "assistente" tem algumas limitações. Em outras palavras, da forma como está, ele não é capaz de explorar todas as possibilidades de uso dos comandos do DOS. Por exemplo, nos micros da linha MSX, ele não usa os caracteres ? e * para listar os arquivos de um determinado tipo, presentes no catálogo.

Se você conhece bem os comandos do DOS do seu micro, não será difícil modificar o programa apresentado, ampliando suas funções e utilidade.

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR LINHA

LINHA	FABRICANTE	MODELO
Apple II +	Appletronica	Thor 2010
Apple II +	CCE	MC-4000 Exato
Apple II +	CPA	Absolutus
Apple II +	CPA	Polaris
Apple II +	Digitus	DGT-AP
Apple II +	Dismac	D-8100
Apple II +	ENIAC	ENIAC II
Apple II +	Franklin	Franklin
Apple II +	Houston	Houston AP
Apple II +	Magnex	DM II
Apple II +	Maxitronica	MX-2001
Apple II +	Maxitronica	MX-48
Apple II +	Maxitronica	MX-64
Apple II +	Maxitronica	Maxitronic I
Apple II +	Microcraft	Craf II Plus
Apple II +	Milmar	Apple II Plus
Apple II +	Milmar	Apple Master
Apple II +	Milmar	Apple Senior
Apple II +	Omega	MC-400
Apple II +	Polymax	Maxxi
Apple II +	Polymax	Poly Plus
Apple II +	Spectrum	Microengenho I
Apple II +	Spectrum	Spectrum ed
Apple II +	Suporte	Venus II
Apple II +	Sycomig	SIC I
Apple II +	Unitron	AP II
Apple II +	Victor do Brasil	Elppa II Plus
Apple II +	Victor do Brasil	Elppa Jr.
Apple IIe	Microcraft	Craft IIe
Apple IIe	Microdigital	TK-3000 IIe
Apple IIe	Spectrum	Microengenho II
MSX	Gradiente	Expert GPC-1
MSX	Sharp	Hotbit HB-8000
Sinclair Spectrum	Microdigital	TK-90X
Sinclair Spectrum	Timex	Timex 2000
Sinclair ZX-81	Apply	Apply 300
Sinclair ZX-81	Engebras	AS-1000
Sinclair ZX-81	Filcres	NEZ-8000
Sinclair ZX-81	Microdigital	TK-82C
Sinclair ZX-81	Microdigital	TK-83
Sinclair ZX-81	Microdigital	TK-85
Sinclair ZX-81	Prologica	CP-200
Sinclair ZX-81	Ritas	Ringo R-470
Sinclair ZX-81	Timex	Timex 1000
Sinclair ZX-81	Timex	Timex 1500
TRS-80 Mod. I	Dismac	D-8000
TRS-80 Mod. I	Dismac	D-8001/2
TRS-80 Mod. I	LNW	LNW-80
TRS-80 Mod. I	Video Genie	Video Genie I
TRS-80 Mod.III	Digitus	DGT-100
TRS-80 Mod.III	Digitus	DGT-1000
TRS-80 Mod.III	Kemitron	Naja 800
TRS-80 Mod.III	Prologica	CP-300
TRS-80 Mod.III	Prologica	CP-500
TRS-80 Mod.III	Sysdata	Sysdata III
TRS-80 Mod.III	Sysdata	Sysdata Jr.
TRS-80 Mod.IV	Multix	MX-Compacto
TRS-80 Mod.IV	Sysdata	Sysdata IV
TRS-Color	Codimex	CS-6508
TRS-Color	Dynacom	MX-1600
TRS-Color	LZ	Color 64
TRS-Color	Microdigital	TKS-800
TRS-Color	Prologica	CP-400

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR NOME

FABRICANTE	MODELO	PAÍS	LINHA
Appletronica	Thor 2010	Brasil	Apple II +
Apply	Apply 300	Brasil	Sinclair ZX-81
CCE	MC-4000 Exato	Brasil	Apple II +
CPA	Absolutus	Brasil	Apple II +
CPA	Polaris	Brasil	Apple II +
Codimex	CS-6508	Brasil	TRS-Color
Digitus	DGT-100	Brasil	TRS-80 Mod.III
Digitus	DGT-1000	Brasil	TRS-80 Mod.III
Digitus	DGT-AP	Brasil	Apple II +
Dismac	D-8000	Brasil	TRS-80 Mod. I
Dismac	D-8001/2	Brasil	TRS-80 Mod. I
Dismac	D-8100	Brasil	Apple II +
Dynacom	MX-1600	Brasil	TRS-Color
ENIAC	ENIAC II	Brasil	Apple II +
Engebras	AS-1000	Brasil	Sinclair ZX-81
Filcres	NEZ-8000	Brasil	Sinclair ZX-81
Franklin	Franklin	USA	Apple II +
Gradiente	Expert GPC1	Brasil	MSX
Houston	Houston AP	Brasil	Apple II +
Kemitron	Naja 800	Brasil	TRS-80 Mod.III
LNW	LNW-80	USA	TRS-80 Mod. I
LZ	Color 64	Brasil	TRS-Color
Magnex	DM II	Brasil	Apple II +
Maxitronica	MX-2001	Brasil	Apple II +
Maxitronica	MX-48	Brasil	Apple II +
Maxitronica	MX-64	Brasil	Apple II +
Maxitronica	Maxitronic I	Brasil	Apple II +
Microcraft	Craft II Plus	Brasil	Apple II +
Microcraft	Caft IIe	Brasil	Apple IIe
Microdigital	TK-3000 IIe	Brasil	Apple IIe
Microdigital	TK-82C	Brasil	Sinclair ZX-81
Microdigital	TK-83	Brasil	Sinclair ZX-81
Microdigital	TK-85	Brasil	Sinclair ZX-81
Microdigital	TK-90X	Brasil	Sinclair Spectrum
Microdigital	TKS-800	Brasil	TRS-Color
Milmar	Apple II Plus	Brasil	Apple II +
Milmar	Apple Master	Brasil	Apple II +
Milmar	Apple Senior	Brasil	Apple II +
Multix	MX-Compacto	Brasil	TRS-80 Mod.IV
Omega	MC-400	Brasil	Apple II +
Polymax	Maxxi	Brasil	Apple II +
Polymax	Poly Plus	Brasil	Apple II +
Prologica	CP-200	Brasil	Sinclair ZX-81
Prologica	CP-300	Brasil	TRS-80 Mod.III
Prologica	CP-400	Brasil	TRS-Color
Prologica	CP-500	Brasil	TRS-80 Mod.III
Ritas	Ringo R-470	Brasil	Sinclair ZX-81
Sharp	Hotbit HB-8000	Brasil	MSX
Spectrum	Microengenho I	Brasil	Apple II +
Spectrum	Microengenho II	Brasil	Apple IIe
Spectrum	Spectrum ed	Brasil	Apple II +
Suporte	Venus II	Brasil	Apple II +
Sycomig	SIC I	Brasil	Apple II +
Sysdata	Sysdata III	Brasil	TRS-80 Mod.III
Sysdata	Sysdata IV	Brasil	TRS-80 Mod.IV
Sysdata	Sysdata Jr.	Brasil	TRS-80 Mod.III
Timex	Timex 1000	USA	Sinclair ZX-81
Timex	Timex 1500	USA	Sinclair ZX-81
Timex	Timex 2000	USA	Sinclair Spectrum
Unitron	AP II	Brasil	Apple II +
Victor do Brasil	Elppa II Plus	Brasil	Apple II +
Victor do Brasil	Elppa Jr.	Brasil	Apple II +
Video Genie	Video Genie I	USA	TRS-80 Mod. I

UM LOGOTIPO PARA CADA MODELO DE COMPUTADOR

INPUT foi especialmente projetado para microcomputadores compatíveis com as sete principais linhas existentes no mercado.
Os blocos de textos e listagens de programas aplicados apenas a determinadas linhas de micros podem ser identificados por meio dos seguintes símbolos:

Sinclair ZX-81

TRS-80

TK-2000

MSX

Spectrum

TRS-Color

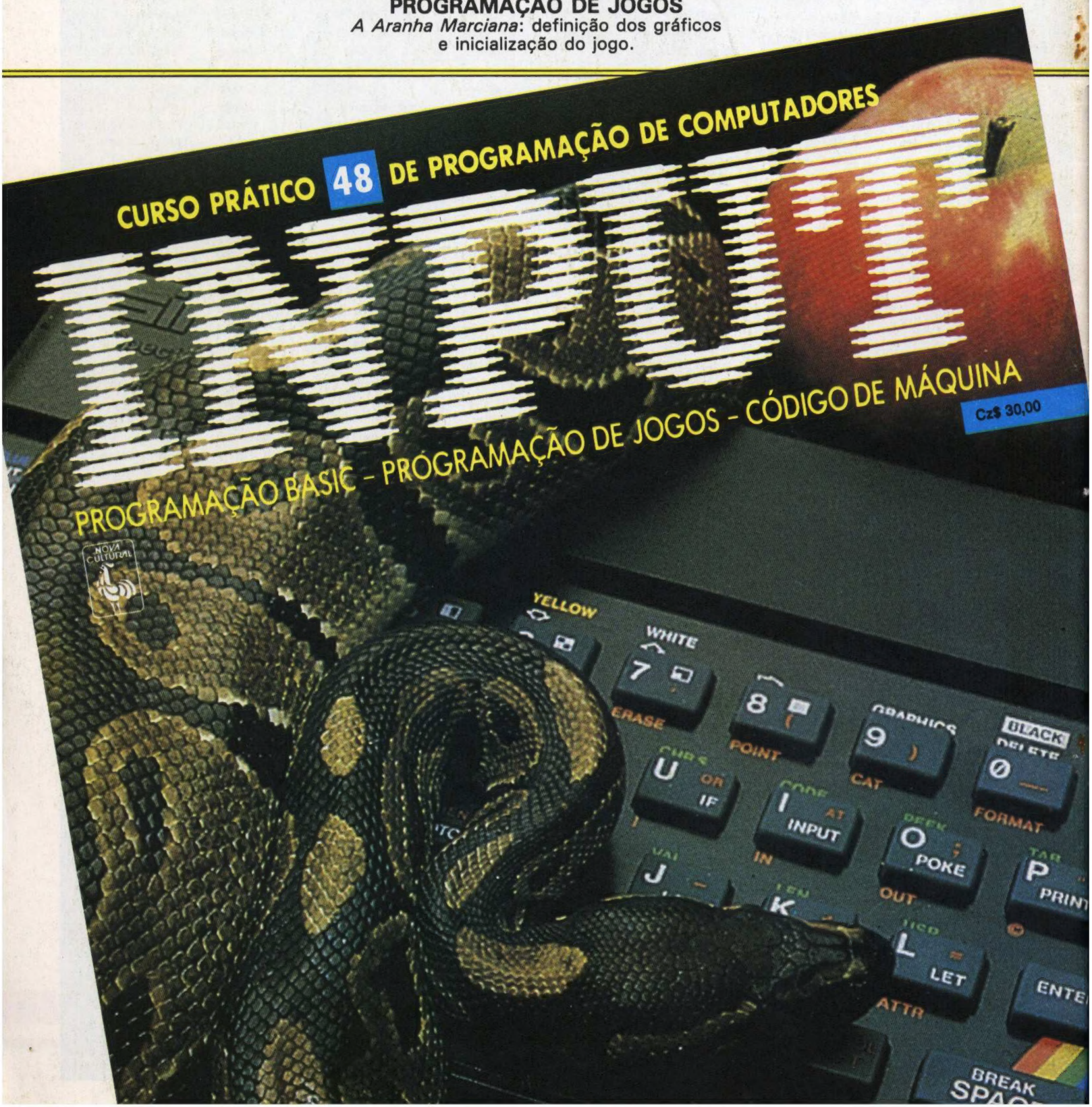
Apple II

Quando o emblema for seguido de uma faixa, então tanto o texto como os programas que se seguem passam a ser específicos para a linha indicada.

NO PRÓXIMO NÚMERO

PROGRAMAÇÃO DE JOGOS
Níveis do jogo *A Raposa e os Gansos*. Rotinas de movimentação da raposa e dos gansos. Tabela de posições.

CÓDIGO DE MÁQUINA
Avalanche: os perigos da aventura. Níveis de dificuldades — os buracos, as cobras e a maré. Rotina de prêmios.

PROGRAMAÇÃO BASIC
A função **VARPTR** no TRS-80. Sub-rotina de transferência. Cópia e impressão da tela.

PROGRAMAÇÃO DE JOGOS
A Aranha Marciana: definição dos gráficos e inicialização do jogo.